공학박사 학위논문

User Experience Enhancement on Smartphones using Wireless Communication Technologies

무선 통신 기술을 활용한 스마트폰에서의 사용자 경험 향상

2020년 8월

서울대학교 대학원

전기 · 컴퓨터공학부

최 준 영

공학박사 학위논문

User Experience Enhancement on Smartphones using Wireless Communication Technologies

무선 통신 기술을 활용한 스마트폰에서의 사용자 경험 향상

2020년 8월

서울대학교 대학원

전기 · 컴퓨터공학부

최 준 영

User Experience Enhancement on Smartphones using Wireless Communication Technologies

지도교수 박 세 웅

이 논문을 공학박사 학위논문으로 제출함

2020년 7월

서울대학교 대학원

전기컴퓨터공학부

최 준 영

최준영의 공학박사 학위 논문을 인준함

2020년 6월

위 원 장: 심 병 효 (인)

부위원장: 박 세 웅 (인)

위 원: 오 정 석 (인)

위 원: 이 경 한 (인)

위 원: 주 창 희 (인)

Abstract

Recently, various sensors as well as wireless communication technologies such as Wi-Fi and Bluetooth Low Energy (BLE) have been equipped with smartphones. In addition, in many cases, users use a smartphone while on the move, so if a wireless communication technologies and various sensors are used for a mobile user, a better user experience can be provided. For example, when a user moves while using Wi-Fi, the user experience can be improved by providing a seamless Wi-Fi service. In addition, it is possible to provide a special service such as indoor positioning or navigation by estimating the user's mobility in an indoor environment, and additional services such as location-based advertising and payment systems can also be provided. Therefore, improving the user experience by using wireless communication technology and smartphone's sensors is considered to be an important research field in the future.

In this dissertation, we propose three systems that can improve the user experience or convenience by using Wi-Fi, BLE, and smartphone's sensors: (i) `BLEND`: BLE beacon-aided fast Wi-Fi handoff for smartphones, (ii) `PYLON`: Smartphone based Indoor Path Estimation and Localization without Human Intervention, (iii) `FINISH`: Fully-automated Indoor Navigation using Smartphones with Zero Human Assistance.

First, we propose fast handoff scheme called `BLEND` exploiting BLE as secondary radio. We conduct detailed analysis of the sticky client problem on commercial smartphones with experiment and close examination of Android source code. We propose `BLEND`, which exploits BLE modules to provide smartphones with prior knowledge of the presence and information of APs operating at 2.4 and 5 GHz Wi-Fi channels. `BLEND` operating with only application requires no hardware and Android source code modification of smartphones. We prototype `BLEND` with commercial smartphones and evaluate the performance in real environment. Our measurement results demonstrate that `BLEND` significantly improves throughput and video bitrate by up to 61% and

111%, compared to a commercial Android application, respectively, with negligible energy overhead.

Second, we design a path estimation and localization system, termed PYLON, which is plug-and-play on Android smartphones. PYLON includes a novel landmark correction scheme that leverages real doors of indoor environments consisting of floor plan mapping, door passing time detection and correction. It operates without any user intervention. PYLON relaxes some requirements for localization systems. It does not require any modifications to hardware or software of smartphones, and the initial location of WiFi APs, BLE beacons, and users. We implement PYLON on five Android smartphones and evaluate it on two office buildings with the help of three participants to prove applicability and scalability. PYLON achieves very high floor plan mapping accuracy with a low localization error.

Finally, We design a fully-automated navigation system, termed FINISH, which addresses the problems of existing previous indoor navigation systems. FINISH generates the radio map of an indoor building based on the localization system to determine the initial location of the user. FINISH relaxes some requirements for current indoor navigation systems. It does not require any human assistance to provide navigation instructions. In addition, it is plug-and-play on Android smartphones. We implement FINISH on five Android smartphones and evaluate it on five floors of an office building with the help of multiple users to prove applicability and scalability. FINISH determines the location of the user with extremely high accuracy with in one step.

In summary, we propose systems that enhance the user's convenience and experience by utilizing wireless infrastructures such as Wi-Fi and BLE and various smartphone's sensors such as accelerometer, gyroscope, and barometer equipped in smartphones. Systems are implemented on commercial smartphones to verify the performance through experiments. As a result, systems show the excellent performance that can enhance the user's experience.

keywords: Wi-Fi Handoff, BLE, Localization, Path Estimation, Dead Reckoning, Simultaneous Localization and Mapping (SLAM), Indoor Navigation, Smartphones

student number: 2014-22586

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Over the last few years, with the dramatic growth of mobile devices such as smartphones and table PCs, wireless communication technologies such as Wi-Fi and Bluetooth Low Energy (BLE) have become an essential and indispensable part of our everyday life, supporting ever increasing demand of users for various types of services. Especially, users use their smartphones while on the move maintaining wireless connection with Wi-Fi and BLE, and there are possibilities to provide better services using wireless communication technologies.

When smartphone users roam while using mobile applications, Wi-Fi handoff between Wi-Fi access points (APs) should be conducted. In addition, if the location of the user is given, location-based service (LBS) such as indoor navigation or targeted area advertisement can be provided to the user. To provide LBS, indoor localization technology should be preceded. Therefore, three different services for user experience enhancement will be introduced in the dissertation.

Fast Wi-Fi handoff: When smartphone users roam while using mobile applications, handoff between Wi-Fi access points (APs) should be conducted in a timely manner in order to provide continuous service. However, we frequently experience that smart-

phone maintains AP connection with very low signal strength even though nearby APs can provide much higher signal strength, which means smartphone does not perform handoff properly. Such undesirable behavior of smartphone, known as sticky client problem [1], has been considered one of the most annoying problems that causes smartphone users to simply deactivate Wi-Fi. To overcome the sticky client problem, smartphone needs to perform fast handoff on time to nearby APs which can provide higher throughput. However, to find nearby APs relying on a single Wi-Fi interface, it is needed to scan all Wi-Fi channels even if there is no available AP, since it has no prior knowledge of when, i.e., exact timing to scan, and where, i.e., operating channel of APs, to scan. Motivated by the fact that the handoff dedicated to smartphone should be supported, we set our goal as designing a practical and simple handoff scheme, which is directly applicable to smartphones.

Indoor localization and navigation: Despite considerable research effort, localization systems are rarely deployed in the real world. One reason is due to device heterogeneity that is difficult to deal with in practice. For instance, the most common approaches for indoor localization are based on Wi-Fi fingerprinting. These approaches work well with RSSI maps and prior knowledge of AP locations. In other words, they should generate an RSSI map for each device. In fact, obtaining AP location information is difficult, and received signal strength indicator (RSSI) map generation is a labor-intensive and time-consuming data gathering process. To address the aforementioned indoor localization problems, we propose a novel path estimation and localization system that works without human intervention. The main idea behind PYLON is to use entrances (i.e., doors) of a building as landmarks under Wi-Fi and BLE infrastructure. Using IMU sensors and RSSIs of Wi-Fi and BLE, the user's smartphone just collects movement data, e.g., step count, step frequency, walking direction, while walking. The server estimates the user's approximate path and performs landmark correction, using the actual floor plan.

1.2 Overview of Existing Approaches

1.2.1 Wi-Fi handoff for smartphones

SyncScan [2] periodically scans Wi-Fi channels to obtain information about nearby APs before handoff. DeuceScan [3] uses spatiotemporal graph to predict the next AP. As a similar approach, D-Scan [4] and Proactive Scan [5] eliminate Wi-Fi scanning delay by actively probing all Wi-Fi channels. However, since above schemes have no prior knowledge of nearby APs, they cannot prevent unnecessary Wi-Fi scanning.

SWIMMING [6] and WGTT [7] focus on fast vehicular handoff. SWIMMING supports seamless Wi-Fi-based Internet access with "group unicast" manner. All APs have to be configured with the same MAC and IP addresses to provide group unicast. WGTT proposes fine-grained AP selection and queue management algorithm in picocell environment. It is difficult to directly apply in practice because additional infrastructure and hardware/software modification of AP are inevitable to implement WGTT.

802.11k [8] allows an associated AP to provide site report of potential candidate APs according to the movement of a mobile station (STA). On the other hand, 802.11r [9] optimizes authentication process to remove authentication delay. When STA enters mobility domain, it finishes authentication process before actual handoff occurs. However, 802.11r is not widely adopted because it focused on enterprise network, which needs infrastructure, and is unsuitable for home deployment [10].

Another approach to seamless handoff makes simultaneous connection with multiple APs in link or transport layer. MultiNet (also known as VirtualWi-Fi [11]) and Juggler [12] allow STA to associate with multiple APs on different Wi-Fi channels in a round robin manner. FatVAP [13] focuses on balancing the traffic load of multiple APs. However, above approaches cannot be implemented in application layer. In Critoru *et al.* [14], STA associates with multiple APs through MPTCP subflow for each AP. Authors argue that MPTCP is available in the iOS 7. However, MPTCP in

iOS uses cellular and a single AP rather than multiple APs.

MultiScan [15] leverages extra Wi-Fi interface to make a connection to a candidate AP before disconnecting the current one. The commercial smartphone has only a single Wi-Fi interface and adding an additional Wi-Fi interface is impractical. In [16, 17], AP is equipped with multiple Wi-Fi interfaces, where one interface is set to operate as an exclusively reserved channel for the Wi-Fi scanning purpose. The STA has to know the reserved channel of candidate AP in advance.

All the above approaches cannot be directly applied to smartphone because they need modified hardware or source code to be implemented on smartphone. In `BLEND`, smartphone can find nearby APs and perform handoff only through the application. No hardware and Android source code modification is required, and `BLEND` can also coexists with other BLE or Classic Bluetooth (BT) connections with smart devices, e.g., smart band or headphone.

1.2.2 Indoor path estimation and localization

Dead Reckoning: Recent developments in micro-electromechanical systems allow multiple sensors, such as accelerometer, magnetometer, and gyroscope, to be integrated into a small inertial sensor module [18]. IMUs, consisting of accelerometers, magnetometers, and gyroscopes, become cheaper and are mounted on handheld devices, especially smartphones.

Since all the information required to estimate human movements [19], such as heading direction, acceleration, and rotation velocity, is recorded on IMU sensors, the smartphone using dead-reckoning tracks the path that a user passes, if the initial location of the user is known [20–22]. For path estimation of the user, the dead-reckoning system uses steps counted by the accelerometer and moves the user's location in the direction of the progress by the step length, determined by the gyroscope or magnetometer. IMU sensor readings are integrated and averaged [23]. However, a significant drawback is error propagation of sensor readings, even a small error magnified

through integration. Complementary approaches leverage virtual landmarks created by the existing infrastructure of Wi-Fi to prevent accumulation of errors [24, 25]. We use landmarks, i.e., doors, to compensate for error accumulation due to drift in sensor readings.

Landmark-Based Localization: As mentioned above, one solution for error propagation of IMU sensors is a landmark-based approach that compensates for drift in sensor readings. Walkie-Markie [24] defines APs as landmarks to fuse crowdsourced user trajectories obtained from IMU sensors on smartphones. However, it requires repeated trajectories of multiple users for the same pathway. JigSaw [26] utilizes crowdsensing images captured from mobile users and extracts the location, size, and orientation information of each landmark object from images. But it is not easy to leverage the images taken by users considering privacy issues. AcMu [27] pinpoints mobile devices with trajectory mapping, and uses them as mobile reference points to collect real-time RSSI samples only when static. It takes days to months for RSSI sample collection to provide accurate localization services. ACMu has limitations in its immediate application to unknown environments. The recent work iVR [28] uses security surveillance cameras installed in an indoor space, and employs a particle filter to fuse data from multiple systems including vision, radio, and IMU sensors. It is more difficult to use video of security surveillance cameras when considering privacy and security issues, compared to using only radio signals of Wi-Fi or BLE. In `PYLON`, we are free from data collection from multiple users, images taken by users, or video from surveillance cameras. `PYLON` can be easily applied to unknown environments.

1.2.3 Indoor navigation

Localization-based Navigation System: Localization and Navigation have been extensively studied in the robotic area [29]. By fusing odometer outputs using IMU sensor readings, robots can compute travel distances, perform accurate localization, and navigate themselves to the destination based on the floor map. In several robotic sys-

tems, additional sensing techniques using laser [30], infrared [31], and camera [32] are also used for ranging and navigation purposes. However, the motion of human is more complicate, and the limited sensing capabilities of smartphones also challenge to both localization and navigation.

To address the aforementioned challenges, numerous localization techniques using smartphones have been proposed [33–40]. They can be broadly categorized as infrastructure-based (e.g, Wi-Fi, BLE, GPS, and cellular) or infrastructure-free (e.g., dead reckoning) localization system. Each of them has its own advantages and disadvantages. GPS can provide accurate positioning in outdoor open spaces but encounters fading signals in an indoor environments. Using radio signals, such as Wi-Fi and BLE, usually requires fingerprinting data to realize localization. In case of dead reckoning, it suffers from cumulative errors and the usage of smartphones.

Peer-to-peer Navigation System: The navigation systems using leader-follower model have been proposed [29, 41–43]. An electronic escort system was proposed by using crowd encounters information and dead-reckoning techniques [41], but it requires pre-deployed audio beacons which limits its applicability. A vision-guided navigation system, termed Travi-Navi [42], enables a user to easily bootstrap and deploy indoor navigation system without help of indoor localization system, but guiders need to hold the smartphone vertically and steadily during walking to achieve a better image quality. FollowMe [29] uses compute-intensive particle filtering as the navigation engine, and minimizes the constraints imposed on users by providing wider usage (i.e., in multi-level buildings, semi-outdoors). FollowUs [43] is incrementally-deployable navigation by automatically generating the trace with the data of multiple users. However, above two navigation systems require assistance of user. The leader needs to input the initial location and destination to provide navigation service, which is vulnerable to the false input of the leaders of malicious users.

1.3 Main Contributions

1.3.1 BLEND: BLE Beacon-aided Fast Handoff for Smartphones.

We envision the approach of BLEND to be increasingly practical and feasible for the following reasons. First, many of today's APs mentioned above are equipped with WPAN modules. Furthermore, legacy APs without BLE modules can be easily connected with BLE module through a USB port. Second, most commercial smartphones are equipped with Bluetooth module. BLE is supported beginning Android 4.3 Jelly Bean launched in 2013, and no hardware modification is required on smartphone. Third, the application-layer solution is possible through user-friendly Android application programming interfaces (APIs) of BLE.

Our major contributions are summarized as follows:

- We conduct detailed analysis of the sticky client problem on commercial smartphones with experiment and close examination of Android source code. Through the comprehensive study, we figure out that the sticky client problem is caused by 1) unacceptably long scanning interval and 2) lack of handoff mechanism.

- We propose BLEND, which exploits BLE modules to provide smartphones with prior knowledge of the presence and information of APs operating at both 2.4 and 5 GHz channels. BLEND operating with only application requires no hardware and Android source code modification of smartphone. To our best knowledge, BLEND is the first BLE-aided handoff scheme.

- We also propose an advanced version of BLEND that can be applied to smartphone enabling hidden Android API, which optimizes scanning through modification of Android source code.

- We prototype BLEND with commercial smartphones and evaluate the performance in real environments. Our measurement results demonstrate that BLEND significantly improves throughput and video bitrate by up to 61% and 111%,

compared to a commercial Android application, respectively, with negligible energy overhead.

1.3.2 PYLON: Smartphone Based Indoor Path Estimation and Localization with Human Intervention.

Unlike other localization systems, PYLON does not require RSSI fingerprint data, locations of Wi-Fi APs and BLE beacons, and the initial location of the user. Moreover, PYLON does not need any additional hardware or software modifications on the smartphone. We have implemented PYLON on five different Android smartphones and evaluated it in an office building. Our experimental results show that PYLON achieves localization performance within an average error of 1.42 m, exploiting floor plan mapping and door passing time detection.

In summary, the main contributions of this paper are threefold:

- We design a path estimation and localization system, termed PYLON, which is plug-and-play on Android smartphones. PYLON includes a novel landmark correction scheme that leverages real doors of indoor environments consisting of floor plan mapping, door passing time detection and correction. It operates without any user intervention.

- PYLON relaxes some requirements for localization systems. It does not require any modifications to hardware or software of smartphones, and the initial location of Wi-Fi APs, BLE beacons, and users. In addition, on-site investigations for fingerprint or trace data collection which are labor-intensive and time-consuming is not necessary. PYLON can be directly applied to unknown indoor environments.

- We implement PYLON on five Android smartphones and evaluate it on two office buildings with the help of three participants to prove applicability and scalability.

PYLON achieves very high floor plan mapping accuracy with a low localization error.

1.3.3 FINISH: Fully-automated Indoor Navigation using Smartphones with Zero Human Assistance.

To bootstrap the indoor navigation services with zero human assistance, we ask the following question: *Can we enable users to easily bootstrap indoor navigation services without any assistance or intervention of users?* In this paper, we provide an answer through the systematic design and implementation of FINISH with the help of a real-world floor plan. Unlike other navigation system, FINISH does not require fingerprint data and assistance of the users. We have implemented FINISH on five different Android smartphones and evaluated it in the whole floors of an office building. Our experimental results show that FINISH achieves 100% initial location accuracy with in one step and provides timely navigation instructions.

In summary, the main contributions of this paper are threefold:

- We design a fully-automated navigation system, termed FINISH, which addresses the problems of existing previous indoor navigation systems. FINISH generates the radio map of an indoor building based on the localization system to determine the initial location of the user.

- FINISH relaxes some requirements for current indoor navigation systems. It does not require any human assistance to provide navigation instructions. In addition, it is plug-and-play on diverse Android smartphones.

- We implement FINISH on five Android smartphones and evaluate it on five floors of an office building with the help of multiple users to prove applicability and scalability. FINISH determines the location of the user with extremely high accuracy with in one step.

1.4 Organization of Dissertation

The rest of the dissertation is organized as follows.

Chapter 2 presents BLEND, BLE beacon-aided fast Wi-Fi handoff for smartphones. First, we conduct preliminary experiment to observe the sticky client problem with smartphones, and figure out causes of the problem. To solve the problem, BLEND exploits BLE to provide smartphones with prior knowledge of the AP information without Wi-Fi scanning, and to realize plug-and-play by installing only the Android application. We prototype BLEND and demonstrate the performance with various experiments.

In Chapter 3, we propose PYLON, smartphone based indoor path estimation and localization without human intervention. PYLON is a path estimation and localization system, which is plug-and-play on Android smartphones. PYLON relaxes some requirements of the current localization systems, and then does not require any human intervention. We implement PYLON and conduct experiments using five different Android smartphones on two office buildings.

Chapter 4 presents FINISH, fully-automated indoor navigation using smartphones with zero human assistance. FINISH generates the radio map of an indoor building based on the localization system to determine the initial location of the user. FINISH relaxes some requirements for current indoor navigation systems. It does not require any human assistance to provide navigation instructions. We implement FINISH on five Android smartphones and evaluate it on five floors of an office building with the help of multiple users to prove applicability and scalability.

Finally, Chapter 5 concludes the dissertation with the summary of contributions and discussion of the future research directions.

Chapter 2

BLEND: BLE Beacon-Aided Fast Wi-Fi Handoff for Smartphones

2.1 Introduction

Over the past few years, the extreme proliferation of smartphones has made Wi-Fi an indispensable wireless technology in our daily lives. Wi-Fi is expected to serve as a hub for immense mobile applications and data traffic in the future [44]. Well-known mobile applications include twittering, video streaming, and Voice over IP, and smartphone users frequently use these applications on the move.

When smartphone users roam while using mobile applications, handoff between Wi-Fi access points (APs) should be conducted in a timely manner in order to provide continuous service. However, we frequently experience that smartphone maintains AP connection with very low signal strength even though nearby APs can provide much higher signal strength, which means smartphone does not perform handoff properly. Such undesirable behavior of smartphone, known as *sticky client problem* [1], has been considered one of the most annoying problems that causes smartphone users to simply deactivate Wi-Fi. To overcome the sticky client problem, smartphone needs to perform fast handoff on time to nearby APs which can provide higher throughput.

In relation to the aforementioned problem, there have been several studies to trigger fast handoff, by utilizing only a single or multiple Wi-Fi interfaces. The first approach exploits only a single Wi-Fi interface to minimize Wi-Fi scanning and handoff delay [2–5, 45]. However, to find nearby APs relying on a single Wi-Fi interface, it is needed to scan all Wi-Fi channels even if there is no available AP, since it has no prior knowledge of when, i.e., exact timing to scan, and where, i.e., operating channel of APs, to scan. The second approach leverages multiple Wi-Fi interfaces. That is, the secondary Wi-Fi interface is dedicated to scanning purpose to reduce scanning delay [16, 17] or pre-association with nearby APs [15]. Using multiple Wi-Fi interfaces is inapplicable to smartphones, since commercial smartphones have only a single Wi-Fi interface. As a result, with the above approaches, there is no way to acquire prior knowledge of nearby APs without Wi-Fi scanning.

However, recent studies show that it is possible to detect nearby APs by utilizing a collocated wireless personal area network (WPAN) radio such as Bluetooth and ZigBee [46–49]. The motivation of the existing work is to minimize energy consumption caused by unnecessary Wi-Fi scanning. Accordingly, as the above approach, WPAN radio can give clue to find nearby APs before handoff and provides the possibility to overcome the limitation of smartphone equipped with a single Wi-Fi interface. In addition, state-of-the-art APs such as Samsung Connect Home [50] and Google Wi-Fi AP [51] are embedded with Bluetooth 4.1 and Zigbee as well as Wi-Fi. Therefore, we expect there is a chance to exploit WPAN.

Motivated by the fact that handoff dedicated to smartphone should be supported, we set our goal as designing a practical and simple handoff scheme, which is directly applicable to smartphones. In this paper, we propose fast handoff scheme called BLEND exploiting Bluetooth Low Energy (BLE) as secondary radio.

Each AP collocated with BLE periodically transmits BLE advertising packet containing its information such as operating channel and channel utilization. Upon an advertising packet reception, smartphone can acquire prior knowledge about exis-

tence and information of nearby APs without Wi-Fi scanning. After that, if handoff is needed, smartphone performs Wi-Fi scanning and estimates throughput using received signal strength indicator (RSSI) and channel utilization. Based on the estimated throughput, smartphone performs handoff to the nearby AP if it can provide better throughput.

We envision the approach of BLEND to be increasingly practical and feasible for the following reasons. First, many of today's APs mentioned above are equipped with WPAN modules. Furthermore, legacy APs without BLE modules can be easily connected with BLE module [52] through a USB port. Second, most commercial smartphones are equipped with Bluetooth module. BLE is supported beginning Android 4.3 Jelly Bean launched in 2013, and no hardware modification is required on smartphone. Third, the application-layer solution is possible through user-friendly Android application programming interfaces (APIs) of BLE.

Our major contributions are summarized as follows:

- We conduct detailed analysis of the sticky client problem on commercial smartphones with experiment and close examination of Android source code. Through the comprehensive study, we figure out that the sticky client problem is caused by 1) unacceptably long Wi-Fi scanning interval and 2) lack of handoff mechanism.

- We propose BLEND, which exploits BLE modules to provide smartphones with prior knowledge of the presence and information of APs operating at both 2.4 and 5 GHz Wi-Fi channels. BLEND operating with only application requires no hardware and Android source code modification of smartphone. To our best knowledge, BLEND is the first BLE-aided handoff scheme.

- We also propose an advanced version of BLEND that can be applied to smartphone enabling hidden Android API, which optimizes Wi-Fi scanning through modification of Android source code.

- We prototype BLEND with commercial smartphones and evaluate the performance in real environments. Our measurement results demonstrate that BLEND significantly improves throughput and video bitrate by up to 61% and 111%, compared to a commercial Android application, respectively, with negligible energy overhead.

2.2 Related Work

2.2.1 Wi-Fi-based Handoff

SyncScan [2] periodically scans Wi-Fi channels to obtain information about nearby APs before handoff. DeuceScan [3] uses spatiotemporal graph to predict the next AP. As a similar approach, D-Scan [4] and Proactive Scan [5] eliminate Wi-Fi scanning delay by actively probing all Wi-Fi channels. However, since above schemes have no prior knowledge of nearby APs, they cannot prevent unnecessary Wi-Fi scanning.

SWIMMING [6] and WGTT [7] focus on fast vehicular handoff. SWIMMING supports seamless Wi-Fi-based Internet access with "group unicast" manner. All APs have to be configured with the same MAC and IP addresses to provide group unicast. WGTT proposes fine-grained AP selection and queue management algorithm in picocell environment. It is difficult to directly apply in practice because additional infrastructure and hardware/software modification of AP are inevitable to implement WGTT.

802.11k [8] allows an associated AP to provide site report of potential candidate APs according to the movement of a mobile station (STA). On the other hand, 802.11r [9] optimizes authentication process to remove authentication delay. When STA enters mobility domain, it finishes authentication process before actual handoff occurs. However, 802.11r is not widely adopted because it focused on enterprise network, which needs infrastructure, and is unsuitable for home deployment [10].

Another approach to seamless handoff makes simultaneous connection with mul-

tiple APs in link or transport layer. MultiNet (also known as VirtualWi-Fi [11]) and Juggler [12] allow STA to associate with multiple APs on different Wi-Fi channels in a round robin manner. FatVAP [13] focuses on balancing the traffic load of multiple APs. However, above approaches cannot be implemented in application layer. In Critoru *et al.* [14], STA associates with multiple APs through MPTCP subflow for each AP. Authors argue that MPTCP is available in the iOS 7. However, MPTCP in iOS uses cellular and a single AP rather than multiple APs.

MultiScan [15] leverages extra Wi-Fi interface to make a connection to a candidate AP before disconnecting the current one. The commercial smartphone has only a single Wi-Fi interface and adding an additional Wi-Fi interface is impractical. In [16, 17], AP is equipped with multiple Wi-Fi interfaces, where one interface is set to operate as an exclusively reserved channel for the Wi-Fi scanning purpose. The STA has to know the reserved channel of candidate AP in advance.

All the above approaches cannot be directly applied to smartphone because they need modified hardware or source code to be implemented on smartphone. In BLEND, smartphone can find nearby APs and perform handoff only through the application. No hardware and Android source code modification is required, and BLEND can also coexists with other BLE or Classic Bluetooth (BT) connections with smart devices, e.g., smart band or headphone.

2.2.2 WPAN-aided AP Discovery

The motivation for exploiting low-power WPAN technologies for Wi-Fi is to reduce large energy consumption caused by unnecessary Wi-Fi scanning. Especially, recent several researches have addressed the problem of finding nearby APs through WPAN. In [46–48], WPAN is exploited to predicts the existence of nearby APs by detecting the signature of periodic Wi-Fi beacon frame through RSSI sampling. These schemes require computational cost and have false positive to detect AP depending on Wi-Fi channel utilization. Since they ceaselessly conduct RSSI sampling, detecting AP is

impossible when WPAN connection exists. In addition, in the case of Zigbee, an external transceiver is needed on smartphone. Blue-Fi [49] predicts nearby APs based on history of heterogeneous network connections and location. However, Blue-Fi is difficult to operate in an unknown environment without history. In conclusion, none of the above approaches deals with fast handoff, and there is no handoff scheme exploiting BLE.

2.3 Background

2.3.1 Handoff Procedure in IEEE 802.11

IEEE 802.11 standard defines handoff procedure as the following three phases: scanning, authentication, and re-association. In scanning phase, STA scans Wi-Fi channels to find APs in vicinity and to acquire information such as RSSI of APs [5]. If STA determines an AP to perform handoff, STA and the AP exchange authentication request and response during authentication phase. After authentication phase, re-association request and response are exchanged during re-association phase, then handoff procedure finishes. Handoff delay represents time between scanning phase and re-association phase. Typically, both authentication and re-association phases are less than 20 ms [2]. The scanning phase empirically accounts for 95% of handoff delay [53] and has been a goal to minimize for fast handoff in literature.

2.3.2 BSS Load Element in IEEE 802.11

IEEE 802.11e [54] defines Basic Service Set (BSS) Load element field, which can be optionally included in beacon and probe response. BSS Load element is composed of station count, channel utilization, and available admission capacity. Station count indicates a total number of STAs currently associated with an AP. Channel utilization is defined as the ratio of time that the AP sensed the medium was busy. Available admission capacity defines remaining amount of time via explicit admission control.

BSS Load element information can be used by STAs when performing handoff.

2.3.3 Bluetooth Low Energy

Bluetooth 4.0 specification [55] defines Classic BT and BLE. BLE is a low-power and short-range wireless communication technology operating on 2.4 GHz unlicensed band. BLE uses 40 physical channels separated by 2 MHz from 2402 MHz to 2480 MHz. Three of 40 channels are used as advertising channels with center frequencies of 2402, 2426, and 2480 MHz. BLE beacon, which represents device equipped with BLE module, periodically broadcasts advertising packet on advertising channels. The interval of advertising packet is in range of 100 ms to 10.24 s.

Advertising packet is composed of preamble, access address, cyclic redundancy check (CRC), and packet data unit (PDU). Preamble is used for detection of advertising packet, synchronization, and automatic gain control. Access address defines the type of BLE packet, e.g., advertising packet. CRC is used to check the validation of advertising packet. PDU includes header field which contains MAC address of BLE beacon, advertising data, and information about advertising packet, e.g., transmission power and packet length.

2.4 Sticky Client Problem

In this section, we experimentally confirm that the sticky client problem occurs on commercial smartphones. Moreover, we analyze the causes of the sticky client problem by investigating experiment results and Android source code [56], especially focusing on Wi-Fi scanning and handoff operations.

2.4.1 Sticky Client Problem of Commercial Smartphone

We conduct an experiment in a controlled office environment where floor plan is illustrated in Fig. 2.1. AP 1 and AP 2 operate on different channels of 5 GHz and trans-

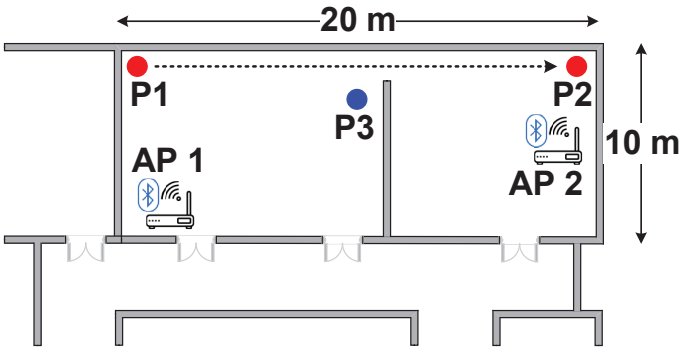
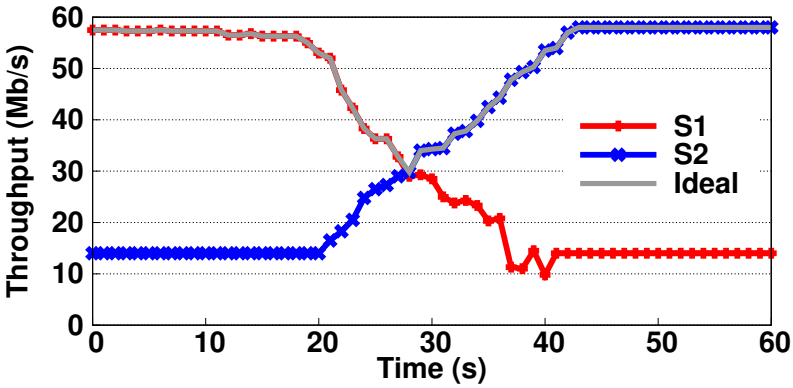


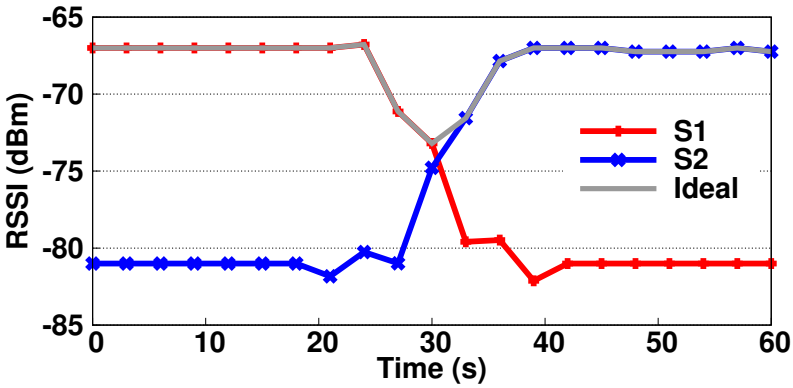
Figure 2.1: Indoor office topology for our experiments. Each point represents locations of smartphone.

mission power of both APs is set to the maximum value of 15 dBm. We use Google Nexus 5 with Android 6.0.1 Marshmallow for the experiment. The smartphone is initially located at P1 for 20 s, moves from P1 to P2 with an average walking speed of 1 m/s, and then stays at P2 for the last 20 s. In scenario 1 (S1) and scenario 2 (S2), smartphone is initially associated with AP 1 and AP 2, respectively. Both APs generate saturated UDP traffic only when smartphone is associated. We measure throughput and RSSI at smartphone every second during 60 s and repeat measurement five times.

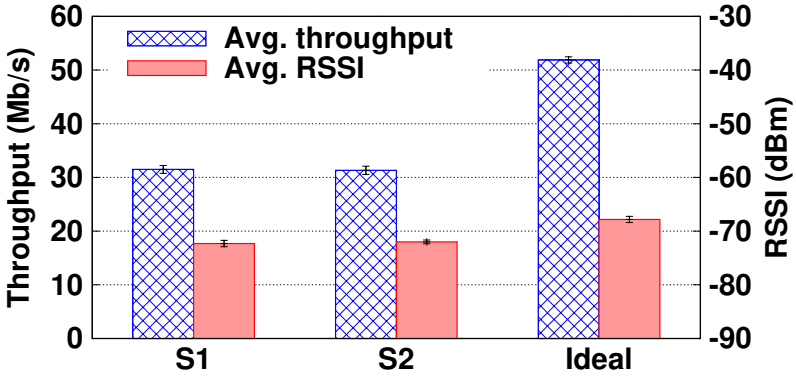
Figs. 2.2a and 2.2b represent average throughput and RSSI in time domain. In S1, throughput and RSSI decrease as smartphone moves away from AP 1. When smartphone reaches P2, we expect that smartphone performs handoff to AP 2. However, smartphone never performs handoff to AP 2 even though RSSI of AP 1 is -81 dBm and RSSI of AP 2 is 15 dBm higher. Similar to above phenomenon, smartphone does not perform handoff to AP 1 at first 20 s in S2. We observe the sticky client problem in both scenarios. The smartphone does not perform handoff even if RSSI of the associated AP is extremely low and there exists another AP which can support higher RSSI. If smartphone performs handoff as Ideal, throughput can be enhanced by 65.7%



(a) Average throughput in time domain.



(b) Average RSSI in time domain.



(c) Average throughput and RSSI during 60 s.

Figure 2.2: Average throughput and RSSI results. Ideal denotes the uppermost curves of S1 and S2 assuming that smartphone performs handoff ideally.

as shown in Fig. 2.2c.

2.4.2 Cause of Sticky Client Problem

To analyze causes of the sticky client problem, we examine conditions to trigger Wi-Fi scanning through measurement and Android source code. Next, we investigate whether smartphone performs handoff when Wi-Fi scanning is triggered.

Scanning operation

To examine Wi-Fi scanning operation in Android, we investigate the source code of Android 6.0.1 Marshmallow. The scanning can be triggered by two different roots: *Wi-FiService* and *startScan*. *Wi-FiService* is the system service of Android, and *startScan* is Android API. Both of them trigger Wi-Fi scanning as an active scanning by default. During the active scanning, STA transmits a probe request and listen for probe responses from APs. *Wi-FiService* starts Wi-Fi scanning when the screen of smartphone turns on. The Wi-Fi scanning interval starts at 40 s, increases to 60, 80, 120, 160, 240, 360 s, and then maintains 360 s. We also double-check that smartphone performs Wi-Fi scanning with the above intervals through packet capture using Airpcap [57]. However, *Wi-FiService* never performs Wi-Fi scanning while the screen turns off. In case of the *startScan*, it triggers Wi-Fi scanning whenever called by Android application manually. Therefore, smartphone depends on *Wi-FiService* for Wi-Fi scanning with long time intervals if *startScan* is not called by application. As a result, Wi-Fi scanning operation of Android is insufficient to support fast handoff on time without *startScan*.

Handoff operation

We conduct an experiment to observe handoff operation of Android smartphones when scanning is manually triggered. Google Nexus 5 and Samsung Galaxy S7 are used for the experiment. The smartphones are located at P1 in Fig. 2.1 and associated with AP 2

for 60 s. We trigger *startScan* by force with the minimum Wi-Fi scanning interval of the API. The minimum Wi-Fi scanning intervals of Nexus 5 and Galaxy S7 are 1 s and 3.5 s, respectively. During the experiment, smartphones do not perform handoff to AP 1 even though average RSSI of AP 2 is -81 dBm while that of AP 1 is -44 dBm. As a result, the sticky client problem cannot be resolved by triggering Wi-Fi scanning, and hence, appropriate handoff operation is absolutely necessary.

2.5 BLEND: Proposed Scheme

It is essential to perform Wi-Fi scanning and handoff at appropriate timing to overcome the sticky client problem. In BLEND, smartphone can explicitly acquire the existence and information about candidate AP through reception of advertising packet. By exploiting acquired information such as the channel number and channel utilization, smartphone determines the best AP to perform handoff by estimating performance improvement. We also proposed an advanced version of BLEND, which selectively scans Wi-Fi channel where a candidate AP exists.

2.5.1 Advantages and Necessities of BLE as Secondary Low-Power Radio

The main concept of BLEND is that BLE beacon collocated with AP periodically transmits advertising packet to notify the existence of AP. Considering that BLE is short-range wireless technology compared with Wi-Fi, receiving advertising packet on smartphone can guarantee that there is AP in the vicinity without Wi-Fi scanning. Therefore, BLE can overcome the limitation that smartphone has no prior knowledge of nearby APs before Wi-Fi scanning caused by single Wi-Fi interface. In addition, information such as channel utilization to help handoff decision is conveyed through the payload of advertising packet. Including the information of AP into the payload of advertising packet is the most important factor to allow BLEND to work in applica-

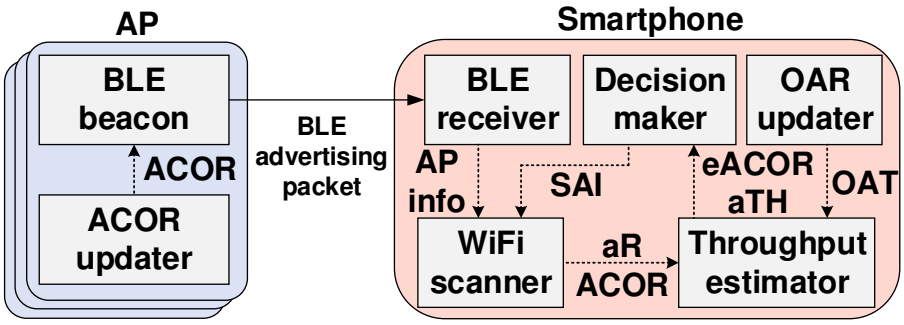


Figure 2.3: Overall architecture of BLEND.

tion layer. The payload of advertising packet can be delivered directly to application layer through BLE Android API. Therefore, BLEND can operate without any additional hardware and Android framework modification. Since BLEND makes no BLE connection, it can operate without interfering other BLE connections, e.g., wearable devices.

In fact, it is possible to design BLEND through only Wi-Fi without BLE. BSS load element including channel utilization can be optionally contained in Wi-Fi beacon. However, using Wi-Fi alone has two drawbacks. First, smartphone should perform Wi-Fi scanning in a periodic or event-driven manner to find nearby APs. Second, even if Wi-Fi beacon contains channel utilization, there is no direct root to deliver information in Wi-Fi beacon to application layer. In order to utilize information in Wi-Fi beacon, firmware modification of smartphone is inevitable. We rule out such approach because it is impractical and not feasible to modify the firmware of all smartphones.

2.5.2 Overall Architecture

Fig. 2.3 shows the overall architecture of BLEND. In AP, available channel occupancy ratio ($ACOR$) is updated in ACOR updater. $ACOR$ conceptually represents channel idle ratio (CIR). BLE beacon periodically transmits advertising packet containing information of AP including $ACOR$ When BLE receiver in smartphone receives advertising packet from BLE beacon, Wi-Fi scanner triggers scanning event depending

on scanning allowance indicator (SAI). SAI adaptively allows Wi-Fi scanner to perform scanning event. Based on RSSI from scanning result, achievable PHY rate (aR) is selected. Throughput estimator calculates achievable throughput (aTH) and effective $ACOR$ ($eACOR$) of AP using $ACOR$, aR, and ongoing airtime ratio (OAR). OAR conceptually represents the ratio of airtime consumed by smartphone itself. Finally, decision maker determines whether to perform scanning and handoff using aTH and $eACOR$, and allows *Wi-Fi scanner* to scan channel depending on SAI.

2.5.3 AP Operation

ACOR updater

To allow smartphone to determine whether or not to perform handoff based on the expected performance improvement, ACOR updater calculates $ACOR$ every second. We first define channel idle ratio (CIR) which represents the ratio of channel idle time during the last 1 s.

$$CIR = 1 - T_{busy}, \tag{2.1}$$

where T_{busy} is channel busy ratio during the last 1 s which includes the time when AP sends and receives packets. To avoid fluctuation of $ACOR$ due to burst traffic, $ACOR$ is defined as the exponentially weighted moving average (EWMA) of CIR as follows:

$$ACOR = \alpha \cdot CIR + (1 - \alpha) \cdot ACOR. \tag{2.2}$$

BLE beacon

BLE beacon takes $ACOR$ from ACOR updater whenever $ACOR$ is updated, and periodically transmits proposed advertising packet. Fig. 2.4 shows proposed advertising packet format. The proposed format basically follows Eddystone beacon format, which is open BLE beacon format from Google [58]. We modify advertising data field to contain information of AP. The proposed advertising data field includes Eddys-

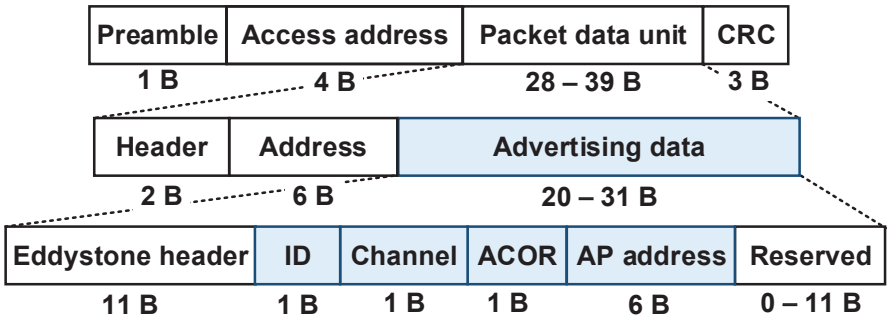


Figure 2.4: Proposed advertising packet of BLEND.

tone header, identifier (ID), channel, $ACOR$, and AP address.[1] Eddystone header is required to maintain Eddystone advertising packet format. ID is needed to verify proposed advertising packet. Channel and AP address indicate the channel number and MAC address of AP, respectively. The rest of data field (11 B) is reserved, which can contain more information about AP.

2.5.4 Smartphone Operation

BLE receiver

It receives advertising packet through monitoring three BLE advertising channels. Upon receiving advertising packet, BLE receiver checks ID field to confirm whether received advertising packet is proposed advertising packet of BLEND. If receiving proposed advertising packet, BLE receiver reads channel, $ACOR$, and AP address fields.

[1]The proposed advertising packet format does not break Eddystone beacon format, and hence, smartphone can receive the proposed advertising packet without defining a new BLE profile.

Table 2.1: IEEE 802.11n PHY rate and Rx sensitivity.

PHY rate (Mb/s)	6.5	13	19.5	26	39	52	58.5	65
Rx sensitivity (dBm)	-94	-91.7	-89.2	-86.1	-82.5	-77.9	-76.3	-74.7

Wi-Fi scanner

It takes AP information whenever BLE beacon receives proposed advertising packets and updates RSSI of AP to determine achievable PHY rate (aR).

RSSI update: Wi-Fi scanner operates in two ways based on AP address. If AP address is the same as MAC address of associated AP (AP_a), Wi-Fi scanner skips scanning because RSSI of AP_a is periodically updated without extra scanning.[2] On the other hand, if AP address is different from AP_a's, it means that candidate AP (AP_c) exists in the proximity. Therefore, Wi-Fi scanner carries Wi-Fi scanning out to update RSSI of AP_c. In the advanced version of BLEND, Wi-Fi scanning is performed in only one channel that AP_c is operating.

Achievable PHY rate: is selected for a given RSSI after updating RSSI of AP. We use receive (Rx) sensitivity values of BCM4339 Wi-Fi chipset [59] for IEEE 802.11n. We employ the Rx sensitivity values of BCM4339 because Nexus 5, a primary smartphone for performance evaluation, is equipped with BCM4339.[3] The Rx sensitivity represents the minimum RSSI value that satisfies 90% packet delivery ratio (PDR) for each PHY rate as shown in Table 2.1. aR is selected as the maximum PHY rate that satisfies PDR 90% for a given RSSI. For example, if RSSI is -76 dBm, 58.5 Mb/s is selected.[4]

[2]RSSI of current associated AP is updated every 3 s in Android by default.

[3]It is also possible to employ different Rx sensitivity values for different smartphone model depending on the embedded Wi-Fi chipset. However, using a common set of values seems to be also acceptable as we observe that the performance is not much affected even if we use a common set for a different smartphone in Section 2.6.

[4]In fact, there is application-level Android API named *getLinkSpeed*, which returns aR in Mb/s of only associated AP. However, we observe that the API cannot reflect real-time aR transition, and hence,

OAR updater

$ACOR$ of AP_a is affected by its own traffic on the smartphone. Throughput of AP_a cannot be determined directly by using $ACOR$ of AP_a. We define ongoing airtime ratio (OAR) which is the ratio of airtime consumed by transmission and reception of itself during the last 1 s. OTA updater calculates OAR every second using EWMA with the same weighting factor (α) as that used for $ACOR$ to maintain consistency between OAR and $ACOR$:

$$OAR = \alpha \cdot \left(\frac{txBits + rxBits}{aR_a} \right) \cdot \frac{1}{t} + (1 - \alpha) \cdot OAR, \tag{2.3}$$

where $txBits$ and $rxBits$ are the number of transmitted and received bits through Wi-Fi during the last t s, respectively, and aR_a is the latest achievable PHY rate of AP_a. We set t to 1 s. Note that OAR updater at smartphone does not need to be synchronized with ACOR updater at AP. In the case of AP_c, OAR is zero because no data is exchanged between AP_c and the smartphone.

Throughput estimator

calculates achievable throughput (aTH) using aR, $ACOR$, and OAR.

Achievable throughput: The operation of throughput estimator differs from AP_a to AP_c. When we estimate aTH of AP_a, OAR should be added to $ACOR$ to reflect the airtime used by smartphone running throughput estimator. We here define effective ACOR ($eACOR$), which is the sum of $ACOR$ and OAR, as follows:

$$eACOR = ACOR + OAR. \tag{2.4}$$

In case of AP_c, $eACOR$ is same as $ACOR$ because OAR of AP_c is zero. After obtaining $eACOR$, aTH is calculated by a product of aR and $eACOR$:

$$aTH = aR \cdot eACOR, \tag{2.5}$$

we use the Rx sensitivity table. Also, the API cannot provide aR of candidate APs.

where aTH conceptually represents the throughput when smartphone fully utilizes the channel during idle time as $eACOR$ with aR. After calculating aTH, Throughput estimator updates both aTH and $eACOR$ of AP.

Decision maker

makes handoff and scanning decisions based on aTH.

Handoff decision: BLEND determines whether to perform handoff to AP_c based on aTH. To avoid ping-pong effect, previous approaches [2, 60] utilize handoff hysteresis (Δ). BLEND also utilizes hysteresis so that smartphone performs handoff if the following condition is satisfied:

$$aTH_a + \Delta < aTH_c, \tag{2.6}$$

where aTH_a and aTH_c are achievable throughput of AP_a and AP_c, respectively. If aTH_c is higher than the sum of aTH_a and Δ, the smartphone performs handoff. In BLEND, Δ is calculated as follows:

$$\Delta = eACOR_a \cdot \delta, \tag{2.7}$$

where $eACOR_a$ is effective $ACOR$ of AP_a and δ is 6.5 Mb/s, which is the minimum throughput difference between adjacent PHY rates as shown in Table 2.1. Therefore, handoff is triggered only when aTH_c is larger than aTH obtained with one step higher achievable PHY rate of AP_a for a given $eACOR_a$.

Scanning decision: Thanks to advertising packet from AP_c, smartphone can explicitly recognize the existence of the AP_c. However, if smartphone triggers scanning event every time it receives advertising packet, unnecessary scanning event can be triggered. Therefore, we define scanning allowance indicator (SAI) to avoid unnecessary scanning event, which is set to true if the following scanning condition (2.8) is satisfied.

$$aTH_a + \Delta < rTH, \tag{2.8}$$

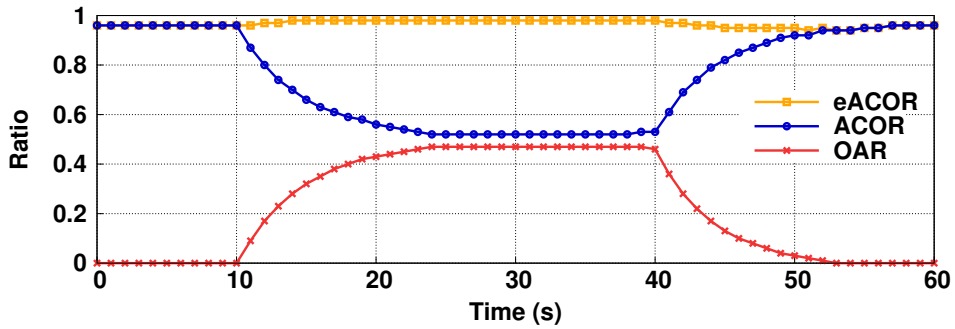
where Δ is the same hysteresis used at handoff decision, and rTH is a reference throughput. Scanning condition implies that scanning event can be triggered only when aTH_a is significantly lower than rTH. It is not desirable to set a constant rTH because the required throughput differs for service type in smartphone. Therefore, BLEND needs to adapt rTH to trigger scanning event depending on the situation, and hence, rTH is controlled by the following two ways.

First, when smartphone triggers scanning event but (2.6) is not satisfied, it means that there is no appropriate AP to perform handoff. In this case, to reduce overhead due to unnecessary scanning, rTH decreases to aTH_a. Note that aTH_a is lower than rTH because scanning condition (2.8) is satisfied. Second, once rTH is set to an extremely low value by the above operation, it is difficult to satisfy (2.8), thus causing a problem of not allowing scanning at all. To avoid such problem, rTH is updated to aTH_a if aTH_a is larger than rTH to allow scanning aggressively for fast handoff when smartphone receives advertising packet of AP_a.

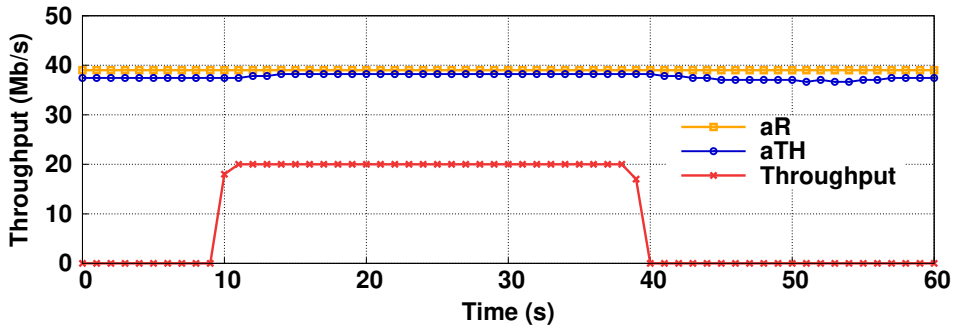
2.5.5 Verification of aTH estimation

We validate aTH estimation process with a simple experiment for 60 s. The experiment is conducted using a smartphone (Nexus 5) without mobility. AP generates 20 Mb/s downlink traffic to the smartphone between 10 s and 40 s. The AP broadcasts advertising packet every 500 ms, and the smartphone receives advertising packet and updates ACOR. Average RSSI of smartphone is -79 dBm.

Fig. 2.5 shows the estimation results. First, aR is selected to be 39 Mb/s since RSSI is -79 dBm (cf. Table 2.1). $ACOR$ and OAR are converged to 0.52 and 0.47, respectively, after traffic generation starts. $eACOR$, the sum of $ACOR$ and OAR, is 0.99, which means the smartphone can occupy up to 99% of channel. However, since $ACOR$ is bound to 0.98 due to Wi-Fi beacon even if there is no data traffic at all, $eACOR$ is overestimated by 1%. Therefore, aTH is 35.1 Mb/s, which is the product of aR and $eACOR$. We identify that aR and aTH estimation work well with 1% error



(a) OAR and $eACOR$ estimation.



(b) aR and aTH estimation.

Figure 2.5: Verification of eACOR and aTH.

of $eACOR$ even if traffic is generated.

2.6 Performance Evaluation

2.6.1 Implementation and Measurement Setup

Implementation: APs are configured by Hostapd-2.5 with Ubuntu 14.04 Linux laptops equipped with Qualcomm Atheros AR9380 chipsets. We modify the latest ath9k device driver, backports-4.2.6-1 [61], to implement ACOR updater. Ubertooth [52], an open source Bluetooth platform equipped with CC2400 transceiver, is attached to each AP for BLE beacon through a USB port. Ubertooth transmits advertising packet every second with the proposed advertising packet format described in Section 2.5.3.[5] The operations of BLEND are implemented as an Android application on smartphone. By default, smartphone performs full scanning, which scans all 2.4 GHz and 5 GHz Wi-Fi channel. For the advanced version of BLEND, direct scanning (DS), which perform Wi-Fi scanning only selected channel, is implemented by enabling Android hidden API, i.e., *customizedScan*. We set α, a weighting factor of EWMA, as 0.2 for $ACOR$ and OAR calculation.

Measurement setup: We conduct our experiments in two scenarios: 1) saturated traffic scenario and 2) video streaming scenario. We conduct all experiments in the topology described in Fig. 2.1. The APs operate at channels 40 and 48 in 5 GHz, respectively, and transmission power is set to 15 dBm. In the saturated traffic scenario, the measurement environment is the same as that in Section 2.4.1. In video streaming scenario, we use ExoPlayer [62] that is an Android open-source video player supporting dynamic adaptive streaming over HTTP (DASH). We encode a 120 s video clip with 16 bitrates from 0.5 Mb/s (320x240) to 15 Mb/s (1920x1080) using FFmpeg [63].[6] All

[5] We set the interval of advertising packet to 1 s, which is the minimum scanning interval of Nexus 5, to avoid redundant transmission of advertising packet.

[6] Full high definition video (1920x1080) can be supported in both Galaxy S7 and Nexus 5.

experiments are repeated five times.

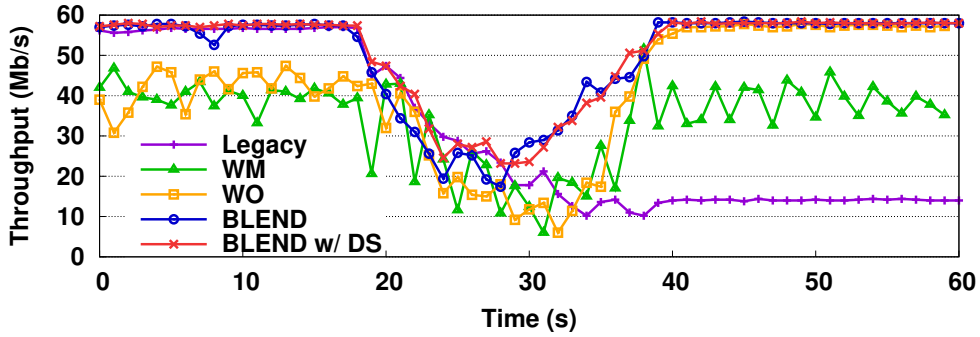
We evaluate BLEND in comparison with the following schemes:

- Legacy: operates without any additional handoff scheme.
- Wi-Fi Manager (WM): with this commercial application [64], smartphone performs scanning event periodically every T interval without condition. When RSSI from the associated AP is under -65 dBm, smartphone performs handoff to a candidate AP supporting 10 dB higher RSSI than the associated AP. To enable fast handoff, we set scanning interval T to 3 s, the minimum configurable value in the application.
- Wi-Fi Only (WO): this home-made application triggers scanning event every T interval only when RSSI is under -65 dBm. We set T to 3 s. Handoff decision is the same as that of WM.
- BLEND w/ DS: the advanced version of BLEND, which performs DS. BLEND w/ DS also works with an application on smartphone by enabling Android hidden API.
- Ideal: the uppermost performance assuming that the smartphone performs handoff ideally.

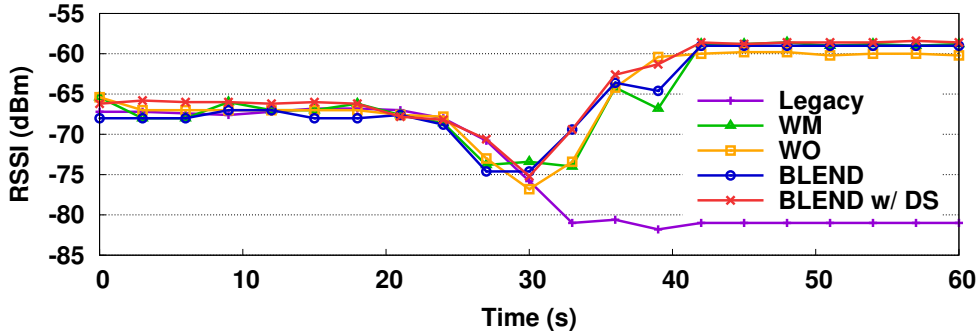
To evaluate the detail operation of BLEND, we use Nexus 5 for saturated traffic scenario. In video streaming scenario, both Nexus 5 and Samsung Galaxy S7 are used to show the device and Android version independence. Due to the unavailability of Android hidden API for Galaxy S7, BLEND w/ DS cannot be implemented in Galaxy S7. We also measure the average energy of Nexus 5 using Monsoon power monitor [65].

2.6.2 Saturated Traffic Scenario

Fast handoff with the help of advertising packet: Fig. 2.6 presents throughput and RSSI results of saturated traffic scenario. Legacy, BLEND, and BLEND w/ DS show



(a) Throughput results for saturated traffic scenario.

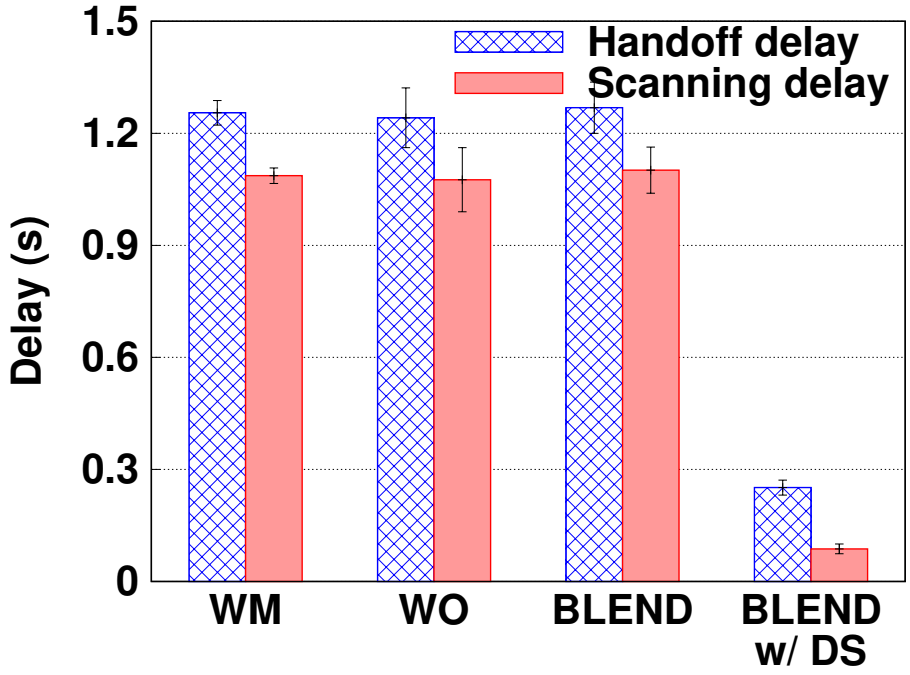


(b) RSSI results for saturated traffic scenario.

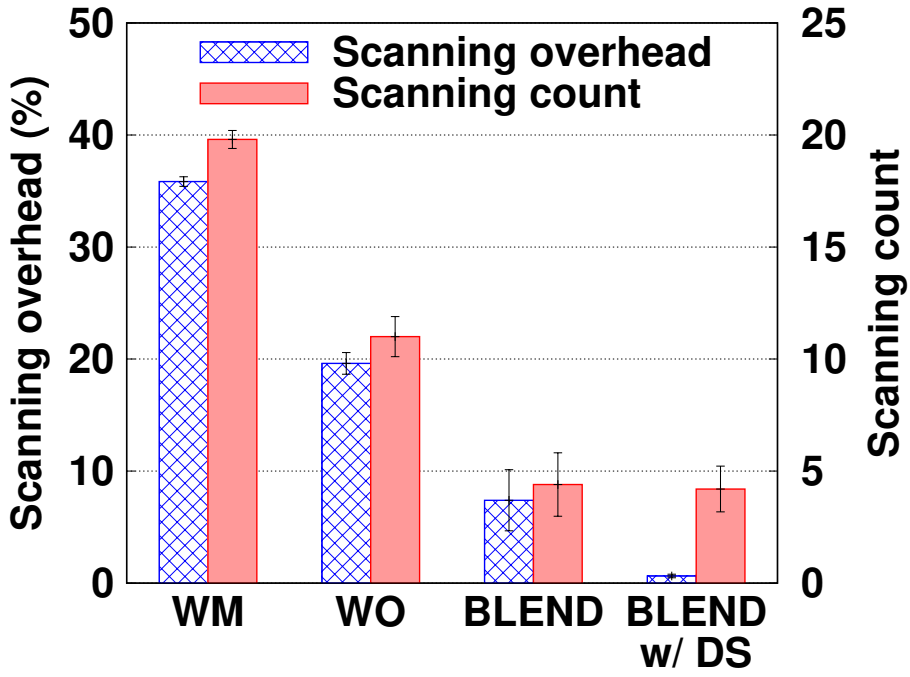
Figure 2.6: Throughput and RSSI results of BLEND and comparison schemes for saturated traffic scenario without background traffic.

the maximum throughput during the first 20 s. However, both WM and WO show degraded and fluctuating throughput due to unnecessary scanning event. WO triggers scanning event every 3 s since RSSI is less than -65 dBm as shown in Fig. 2.6b. After 20 s, throughput values of all the schemes decrease as the smartphone moves away from the associated AP. Throughput and RSSI of Legacy continuously decrease since Legacy does not perform handoff to AP 2 even if RSSI goes under -80 dBm. Throughput of WM still fluctuates even after handoff because it periodically triggers scanning event regardless of RSSI. In cases of BLEND, BLEND w/ DS, and WO, smartphone performs handoff to AP 2 and show the maximum throughput after 40 s. We also observe that BLEND and BLEND w/ DS perform handoff to AP 2 earlier than WM and WO. With the help of advertising packet sent by AP 2, BLEND and BLEND w/ DS make a decision to trigger scanning event every 1 s, thus enabling fast handoff without unnecessary scanning event. This fast handoff enhances throughput of BLEND and BLEND w/ DS than that of WO and WM between 30 s and 40 s.

Handoff and scanning performance: We evaluate BLEND and comparison schemes in terms of delay and overhead. Fig. 2.7a shows average delay per scanning and handoff. As explained in Section 2.3, handoff delay is the sum of scanning, authentication, and re-association delay. Each delay is measured based on state transitions of *Wi-FiManager*, the primary API to manage all aspects of Wi-Fi connectivity in application layer. Scanning delay is calculated as time difference between the start time of scanning event, i.e., the time when *StartScan()* function is called at application, and the time when scanning event finishes. Handoff delay is calculated as time difference between the start time of scanning event and the time when handoff from AP 1 to AP 2 is completed. Since WM, WO, and BLEND perform full scanning, scanning and handoff delay have no difference. However, BLEND w/ DS can perform scanning only for AP 2's channel with the help of advertising packet, and hence, BLEND w/ DS significantly reduces scanning delay by up to 92%. Considering authentication and re-association delay, BLEND w/ DS can reduce handoff delay up to 81% compared to



(a) Handoff and scanning delay.



(b) Scanning overhead and count.

Figure 2.7: Handoff and scanning performance of handoff schemes.

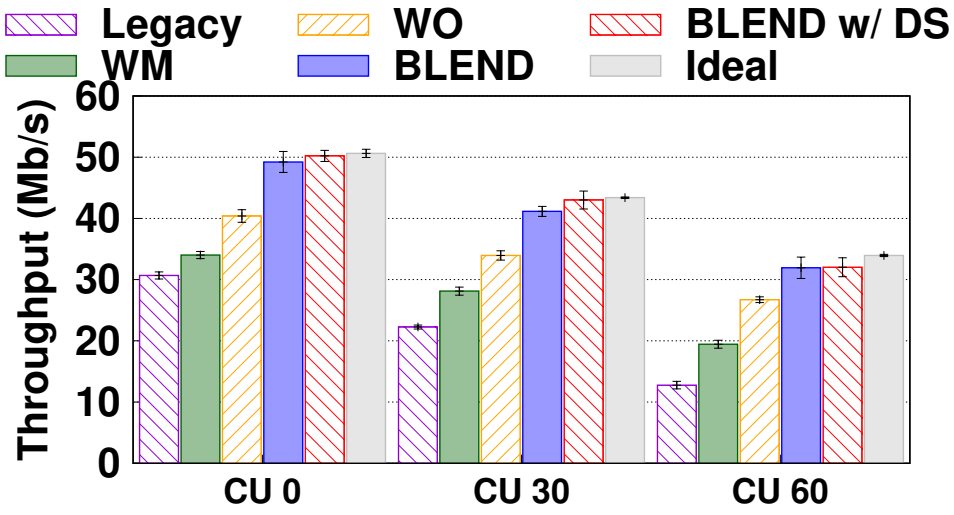
other schemes.

Fig. 2.7b shows average scanning overhead and scanning count during measurements. Scanning overhead denotes the ratio of total scanning time to 60 s, and scanning count is the number of scanning events during 60 s. WM always triggers scanning events every 3 s, thus resulting in the largest scanning overhead and count. WO continues scanning events until performing handoff to AP 2, and hence, scanning overhead is 19.6%. However, BLEND and BLEND w/ DS reduce scanning overhead to 7.8% and 0.6%, respectively, because they trigger scanning only when handoff is required. In case of BLEND w/ DS, scanning overhead is drastically reduced thanks to DS.

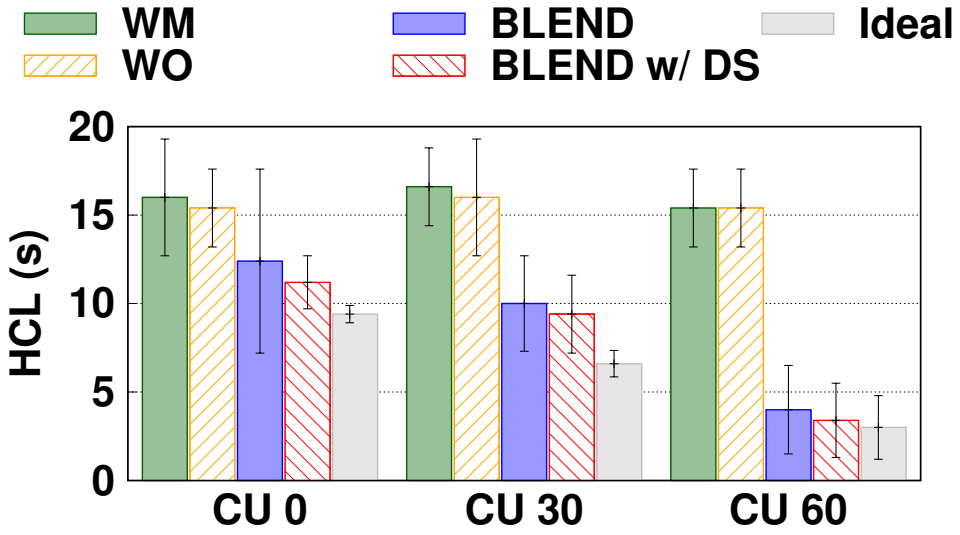
Various channel utilization: For various channel utilization (CU) of AP 1's channel, we generate background UDP traffic. CU 0, CU 30, and CU 60 denote that smartphone can utilize maximum 100%, 70%, and 40% of AP 1's channel, respectively. Fig. 2.8 shows average throughput and handoff completion latency (HCL) under various CU. HCL represents difference between the time when smartphone starts to move (20 s) and the time when smartphone finishes handoff. When smartphone cannot fully utilize AP 1's channel due to the background traffic, it is advantageous to perform handoff to AP 2 more quickly. Therefore, HCL of Ideal decreases as CU increases. Thanks to throughput estimator, HCL of BLEND also decreases as CU increases, and hence, BLEND can achieve average throughput close to Ideal in every CU. Note that WM and WO do not consider CU, HCL of WM and WO do not decrease as CU increases. BLEND can achieve higher throughput than WM and WO up to 61% and 24%, respectively. BLEND w/ DS shows similar results to BLEND, but BLEND w/ DS achieves slightly higher average throughput than BLEND due to DS. Legacy does not perform handoff, and hence, its throughput is less than that of all other schemes.

2.6.3 Video Streaming Scenario

We evaluate the performance of BLEND and measure energy consumption during video streaming. The smartphone is initially associated with AP 1 and stays at P3.



(a) Average throughput.



(b) Average HCL.

Figure 2.8: Experiment results for three different channel utilization.

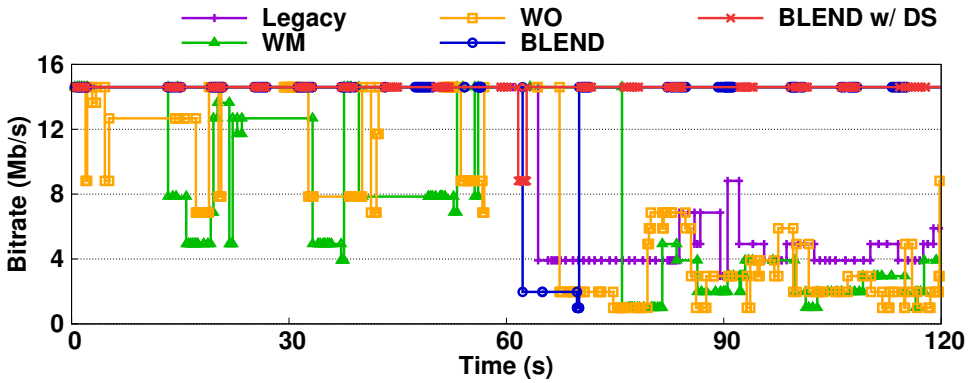


Figure 2.9: The snapshot of video bitrate for Nexus 5.

Average RSSI of both AP 1 and AP 2 is equally -70 dBm at P3. There is no traffic on both channel of APs at first, but saturated background traffic is generated only on AP 1's channel after 60 s, which means CU of AP 1's channel is almost 100.

Snapshot of video bitrate: Fig. 2.9 shows a time snapshot of video bitrate from Nexus 5. Legacy can support the maximum video bitrate for the first 60 s, but bitrate drops sharply during the next 60 s due to background saturated traffic. In case of WM and WO, video bitrate is even lower than Legacy due to unnecessary Wi-Fi scanning. WM and WO do not take into account CU for handoff, and hence, smartphone does not perform handoff to AP 2. On the other hand, with BLEND and BLEND w/ DS, smartphone performs handoff quickly when background saturated traffic is generated, and hence, the maximum video bitrate is supported for the most of time. Since the interruption of video traffic of BLEND w/ DS is shorter than BLEND during handoff due to DS, BLEND w/ DS recovers to the maximum bitrate within 1 s much faster than that of BLEND, i.e., 7 s.

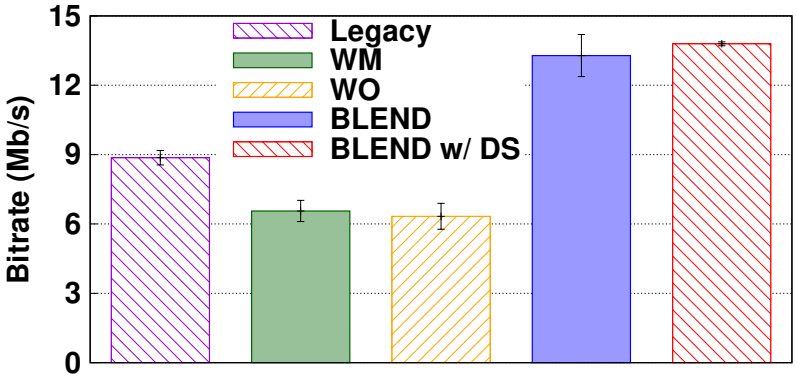
Average bitrate and energy consumption measurement: Figs. 2.10a and 2.10b show average video bitrate and energy consumption of Nexus 5. Legacy shows the lowest energy consumption because there is no Wi-Fi scanning at all. WM and WO consume 0.44 mWh and 0.67 mWh more energy, respectively, compared to Legacy due to

unnecessary Wi-Fi scanning. In addition, WM and WO show lower video bitrate compared to that of Legacy because the interruption of video traffic due to Wi-Fi scanning. However, thanks to fast handoff and intelligent Wi-Fi scanning operation of BLEND, it achieves 104% and 111% higher video bitrate compared to WM and WO, respectively. BLEND w/ DS shows 0.5 Mb/s higher bitrate compared to that of BLEND due to DS. The reason that the energy consumption of BLEND and BLEND w/ DS is 0.67 mWh and 0.63 mWh higher than that of Legacy is playing higher video bitrate consumes more energy. However, it is a negligible cost to support almost maximum video quality.

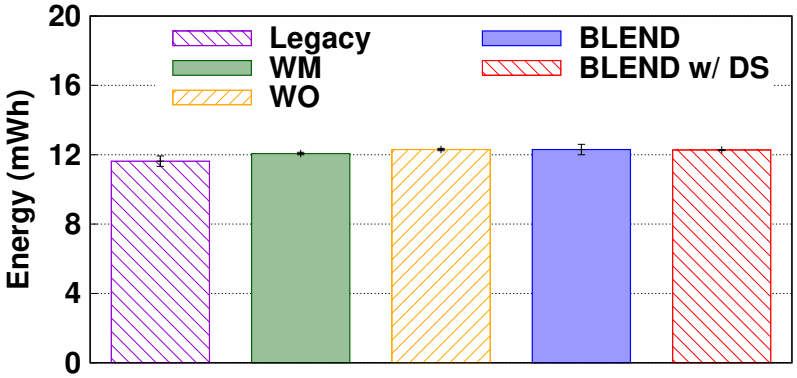
Device independence: Fig. 2.10c presents average video bitrate from Galaxy S7. Because BLEND w/ DS requires enabling Android hidden API, experiment is conducted without BLEND w/ DS in Galaxy S7. Galaxy S7 shows 9.0 Mb/s, which is almost same as Nexus 5. BLEND shows the highest average bitrate of 12.6 Mb/s, which is 173% and 152% higher than WM and WO, respectively. The average bitrate on Galaxy S7 excluding Legacy is lower than that of Nexus 5 because Galaxy S7 has two times longer Wi-Fi scanning delay compared to Nexus 5.

2.7 Summary

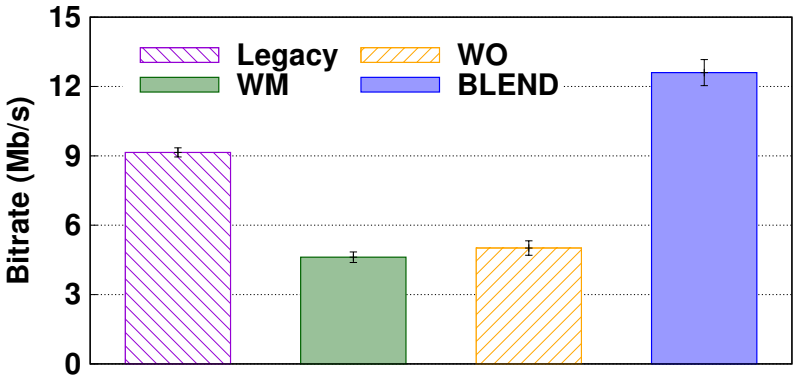
We have proposed BLEND, a novel and practical fast handoff scheme for smartphone in IEEE 802.11, based on the in-depth investigation of smartphones' Wi-Fi scanning and handoff operation. BLEND enables fast handoff to candidate AP by solving the sticky client problem. Our solution is aided by BLE beacon that broadcasts collocated AP's information with advertising packet. We have implemented a prototype as an Android application. With BLEND, smartphone finds target AP and performs fast handoff based on throughput estimation, handoff decision, and adaptive scanning. Our experiment results confirm the practicality, feasibility, and performance of BLEND in diverse environments. We observe that BLEND enhances throughput and video bitrate by up



(a) Average video bitrate of Nexus 5.



(b) Average energy consumption of Nexus 5.



(c) Average video bitrate of Galaxy S7.

Figure 2.10: Experiment results in video streaming and energy measurement: BLEND shows always the highest video bitrate without significant energy overhead.

to 61% and 111%, respectively, compared to the commercial Android application. As future work, we plan to reflect coexistence with legacy AP, which has no BLE module and design more fine-grained throughput estimation algorithm reflecting multiple spatial streams and channel bandwidth.

Chapter 3

PYLON: Smartphone based Indoor Path Estimation and Localization without Human Intervention

3.1 Introduction

Recently, indoor location-based services (LBS) and applications have attracted a lot of attention due to their social and commercial values, with the market value expected to reach US$10 billion by 2020 [66]. Generally, the service quality of LBS depends on the accuracy of location estimation for a user. Therefore, it is very important to precisely estimate the user's location, as well as to track the path the user has traveled. For example, in markets, sales can be increased by placing popular items in locations where many people pass by. In office buildings, we can use users' route information to estimate hotspot candidates, and relocate or increase Wi-Fi access points (APs) to provide better Wi-Fi services. LBS benefits from path estimation of users to provide better services [24].

Despite considerable research effort, localization systems are rarely deployed in the real world. One reason is due to device heterogeneity that is difficult to deal with in practice. For instance, the most common approaches for indoor localization are based on Wi-Fi fingerprinting. These approaches work well with RSSI maps and prior

knowledge of AP locations. In other words, they should generate an RSSI map for each device [67–69]. In fact, obtaining AP location information is difficult, and received signal strength indicator (RSSI) map generation is a labor-intensive and time-consuming data gathering process.

Some recent efforts aim to alleviate the pain of RSSI map construction by exploiting inertial measurement units (IMUs), including accelerometers, gyroscopes, and magnetometers, that are mounted on most smartphones today [35, 36]. These readings represent movement characteristics of a user and help the smartphone to track the walking path of the user, i.e., dead reckoning. However, there are some problems with dead reckoning using IMU sensors. First, sensor drift accumulates quickly over time, so tracking accuracy lasts only for a short time. Second, dead reckoning tracks the walking path, not the exact location of the user. To determine the exact location, knowing the user's initial location is essential, so human intervention is inevitable.

To address the aforementioned indoor localization problems, we propose a novel path estimation and localization system, called PYLON, that works without human intervention. The main idea behind PYLON is to use entrances (i.e., doors) of a building as landmarks under Wi-Fi and Bluetooth Low Energy (BLE) infrastructure. PYLON is an application, running on a smartphone and a back-end server. Using IMU sensors and RSSIs of Wi-Fi and BLE, the user's smartphone just collects movement data, e.g., step count, step frequency, walking direction, while walking. The server estimates the user's approximate path and performs landmark correction, using the actual floor plan.

The central component of PYLON is landmark correction. It is a novel way to take advantage of widely-deployed Wi-Fi and BLE infrastructure, enabling the user to get accurate path estimation and localization. PYLON generates virtual rooms where the user is considered to have passed. It uses a data mining approach to generate virtual rooms according to collected RSSIs. Our proposed floor plan mapping algorithm maps the generated virtual rooms to the real-world floor plan. Then PYLON estimates rooms that the user has actually passed through. After conducting the floor plan map-

ping, PYLON estimates the time when the user passed through each door, using the real-world floor plan. The path roughly generated by IMU sensors is corrected by the doors that the user passed through. Leveraging doors as landmarks, PYLON eliminates cumulative errors in IMU sensors such as location and walking direction, and traces roughly generated paths without human intervention. In addition, PYLON applies a particle filter to increase the accuracy of path estimation and localization.

Unlike other localization systems, PYLON does not require RSSI fingerprint data, locations of Wi-Fi APs and BLE beacons, and the initial location of the user. Moreover, PYLON does not need any additional hardware or software modifications on the smartphone. We have implemented PYLON on five different Android smartphones and evaluated it in an office building. Our experimental results show that PYLON achieves localization performance within an average error of 1.42 m, exploiting floor plan mapping and door passing time detection.

In summary, the main contributions of this paper are threefold:

- We design a path estimation and localization system, termed PYLON, which is plug-and-play on Android smartphones. PYLON includes a novel landmark correction scheme that leverages real doors of indoor environments consisting of floor plan mapping, door passing time detection and correction. It operates without any user intervention.

- PYLON relaxes some requirements for localization systems. It does not require any modifications to hardware or software of smartphones, and the initial location of Wi-Fi APs, BLE beacons, and users. In addition, on-site investigations for fingerprint or trace data collection which are labor-intensive and time-consuming is not necessary. PYLON can be directly applied to unknown indoor environments.

- We implement PYLON on five Android smartphones and evaluate it on two office buildings with the help of three participants to prove applicability and scalability.

PYLON achieves very high floor plan mapping accuracy with a low localization error.

3.2 Background and Related Work

3.2.1 Infrastructure-Based Localization

The lack of compatibility of GPS for indoor localization has led to diverse approaches that use alternative systems such as camera [70], acoustic [71–74], and infrared [75]. These systems require the deployment of infrastructure or specially designed hardware to realize localization for indoor environments. While each approach has its own advantages (e.g., sub-meter level localization accuracy in the case of acoustic systems), requiring specific hardware or infrastructure is a barrier to the wide adoption of these localization systems [38]. SpotFi [76] uses CSI to calculate the angle of arrival (AOA) of multipath components to achieve decimeter-level accuracy. LiFS [77] achieves passive localization by taking advantage of the shadow effect caused by people blocking the line-of-sight paths of the Wi-Fi links. A recent approach proposes a localization system [78] based on RSSI of the 60 GHz wireless LAN, also known as mmWave, compliant with the IEEE 802.11ad standard [79]. However, since mm-Wave devices and APs are not yet widely deployed, it is hard to directly apply mmWave-based localization to the current infrastructure.

In addition to the mmWave-based localization, visible light localization systems have been proposed recently. These systems use the frequency of a measured light with camera [80] or measure the light intensity through ambient light sensors built on smartphones [81–83]. However, light-based localization is not compliant to traditional fluorescent lights in the building, and requires additional hardware to apply to light emitting diodes (LED). Since the measured light intensity is used for localization, it is difficult to accurately specify a position when there exists strong ambient light interference (e.g., sunlight during the daytime). To address the aforementioned

limitations of infrastructure-based localization, we propose PYLON, focusing on applicability. We take advantage of wide deployment of Wi-Fi and BLE to increase the cost-effectiveness of localization systems, and design PYLON that works plug-and-play on smartphones, thereby improving practicality significantly.

3.2.2 Fingerprint-Based Localization

Many indoor localization schemes adopt fingerprint approaches to determine the user's location [68, 69, 84–87]. The localization system in [36], based on measuring and matching radio signals of Wi-Fi or Bluetooth, leverages the already deployed infrastructure to provide significant cost benefits. However, fingerprint-based localization requires a labor intensive work to measure RSSI at every location in the points of interest, and then it should build a fingerprint database. This approach is pioneered by Radar [67] and Horus [88] that identify different causes for the wireless channel variation, resulting in better accuracy with lower complexity. SurroundSence [89] builds a database based on ambience fingerprinting such as light, color, Wi-Fi, etc.

The above approaches require site investigation of a specific area to create a fingerprint database comprising RSSI measurements at known locations. The fingerprint-based localization lacks flexibility in dynamic environments (e.g., variation of AP placement), and hence database construction has to be repeated at each new point to maintain the system. However, PYLON does not require any site investigation and it is flexible to environmental changes, which eliminates the drawbacks of the fingerprint-based system.

3.2.3 Model-Based Localization

An alternative way, which avoids a labor intensive task of fingerprint-based localization, is to use a radio propagation model to estimate the RSSI at a given location x, according to the transmit power P_t and the distance d_x between the transmitter and a given location [90]. A popular model is the log-distance path-loss model that calcu-

lates the RSSI as $P_x = P_t - \gamma log(d_x) + N$, where N represents the noise term [91]. These approaches sacrifice localization accuracy to eliminate database construction efforts. The irregular signal propagation in indoor environments caused by obstacles (e.g., walls) decreases location accuracy, and hence the extension of this model has incorporated the irregularity of signal propagation [33, 67, 88, 92, 93]. These approaches show the limitations in localization accuracy, due to the inaccuracy of models [94]. PYLON does not require any prior knowledge of locations of users or APs.

3.2.4 Dead Reckoning

Recent developments in microelectromechanical systems allow multiple sensors, such as accelerometer, magnetometer, and gyroscope, to be integrated into a small inertial sensor module [18]. IMUs, consisting of accelerometers, magnetometers, and gyroscopes, become cheaper and are mounted on handheld devices, especially smartphones. Since all the information required to estimate human movements [19], such as heading direction, acceleration, and rotation velocity, is recorded on IMU sensors, the smartphone using dead-reckoning tracks the path that a user passes, if the initial location of the user is known [20–22]. For path estimation of the user, the dead-reckoning system uses steps counted by the accelerometer and moves the user's location in the direction of the progress by the step length, determined by the gyroscope or magnetometer. IMU sensor readings are integrated and averaged [23]. However, a significant drawback is error propagation of sensor readings, even a small error magnified through integration. Complementary approaches leverage virtual landmarks created by the existing infrastructure of Wi-Fi to prevent accumulation of errors [24, 25]. We use landmarks, i.e., doors, to compensate for error accumulation due to drift in sensor readings.

3.2.5 Landmark-Based Localization

As mentioned above, one solution for error propagation of IMU sensors is a landmark-based approach that compensates for drift in sensor readings. Walkie-Markie [24] defines APs as landmarks to fuse crowdsourced user trajectories obtained from IMU sensors on smartphones. However, it requires repeated trajectories of multiple users for the same pathway. JigSaw [26] utilizes crowdsensing images captured from mobile users and extracts the location, size, and orientation information of each landmark object from images. But it is not easy to leverage the images taken by users considering privacy issues. AcMu [27] pinpoints mobile devices with trajectory mapping, and uses them as mobile reference points to collect real-time RSSI samples only when static. It takes days to months for RSSI sample collection to provide accurate localization services. ACMu has limitations in its immediate application to unknown environments. The recent work iVR [28] uses security surveillance cameras installed in an indoor space, and employs a particle filter to fuse data from multiple systems including vision, radio, and IMU sensors. It is more difficult to use video of security surveillance cameras when considering privacy and security issues, compared to using only radio signals of Wi-Fi or BLE. In `PYLON`, we are free from data collection from multiple users, images taken by users, or video from surveillance cameras. `PYLON` can be easily applied to unknown environments.

3.2.6 Simultaneous Localization and Mapping (SLAM)

In robotics and computer vision communities, simultaneous localization and mapping (SLAM) is developed as a technique for jointly estimating the location of a robot and the map of an environment [95, 96]. While a robot equipped with sensors, such as camera, sonar, and radar, has the capability of navigating an area of interest, the limitations of a smartphone hinder use of the SLAM technique [97–99]. To realize SLAM, robot sensors detect landmarks or obstacles, and the robot helps to illustrate the discovered area map according to the sensing data. However, it is impractical to

adopt standard SLAM for smartphone-based localization [25].

While early research on SLAM uses Kalman Filter for localization, it applies only to the linearly modeled position of a robot that follows a Gaussian distribution. To apply the robot's position for a nonlinear and non-Gaussian distribution, Monte Carlo Localization (MCL) [100] presents a version of Markov localization as part of SLAM. MCL, also known as particle filter localization, applies sampling-based methods to approximate the probability distribution. It uses particles to represent possible states. Whenever the robot moves, MCL shifts particles to predict new state and re-samples according to recursive Bayesian estimation. Finally, the particles converge to the actual location.

There have been efforts leveraging IMU sensors and radio signals to apply SLAM. FootSLAM [101] uses foot-mounted IMU sensors to construct an inertial map. Wi-Fi-SLAM [102] uses a Gaussian process latent variable model to build an RSSI map and models human movement as a hidden variable. Semi-supervised localization [103] estimates the location of other users according to RSSI dissimilarities. Zee [36] reduces RSSI map generation efforts with the help of the real-world floor plan. LiFS [84] leverages the floor plan and the relationship between rooms including the door, which are the overlapped components of `PYLON`. LiFS requires RSSI samples of the area of interest at the intersection, and the training phase is essential to match the floor plan and raw RSSI data. `PYLON` does not require fingerprint data for the training process. Instead, the real-world floor plan and only RSSI data collected during the user's walk are used.

3.3 System Overview

`PYLON` is performed on two entities: a smartphone and a server as shown in Figure 3.1. Data collection module and data processing module operate on the smartphone. Data collection module collects RSSIs of Wi-Fi/BLE and reads IMU sensors such as gy-

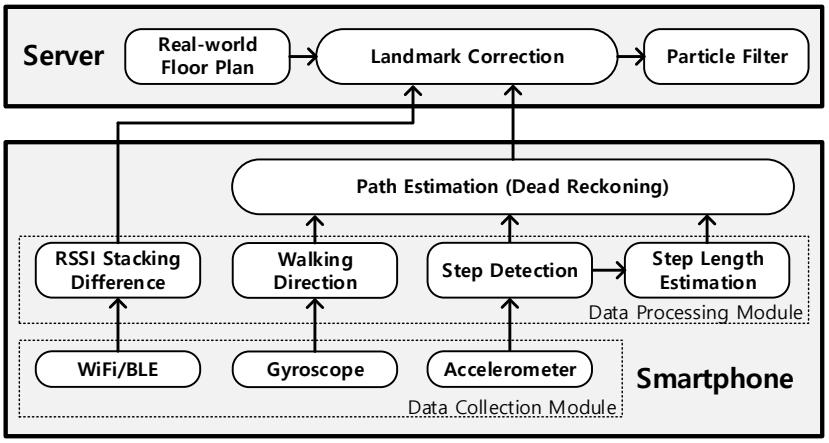


Figure 3.1: System architecture of PYLON.

roscope and accelerometer. Data processing module calculates RSSI differences of Wi-Fi/BLE and roughly generates the route of a user based on walking direction estimation, step detection, and step length estimation by using IMU sensor readings. Landmark correction is conducted to estimate the path and location of the user with the help of the real-world floor plan, and PYLON applies the particle filter on the server to improve estimation accuracy of the path and location.

3.3.1 Notable RSSI Signature

PYLON applies landmark correction to estimate the path and location of the user. Investigating the spatial and temporal characteristics of Wi-Fi and BLE signals, we find an interesting wireless radio signal characteristic that could be used in indoor localization. The major observation is that received signal strength significantly decreases or increases while the user passes through a door. As shown in Figure 3.2, the RSSI of the same AP sharply changes as the user passes through the door between two adjacent rooms at 10 s. Therefore, the characteristics of RSSI can be used to detect the exact timing of the user passing through a door and to differentiate between two different rooms.

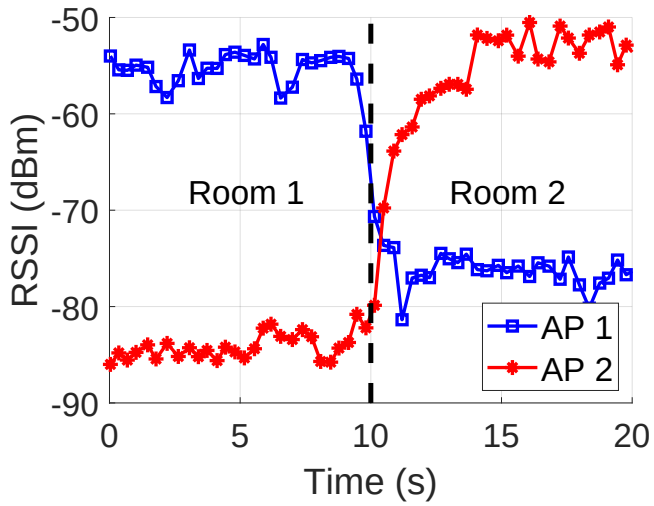


Figure 3.2: Sharp RSSI change when the user passes the door.

3.3.2 Smartphone Operation

Data Collection Module. We use radio devices (i.e., Wi-Fi and BLE) and IMU sensors (i.e., accelerometer and gyroscope) of a smartphone for data collection. The smartphone triggers Wi-Fi scanning periodically to receive packets from multiple APs and to collect RSSIs [104]. BLE beacons transmit advertising packets periodically, and the smartphone uses the received advertising packets to collect RSSIs when its Bluetooth module is on. That is, RSSIs of Wi-Fi and BLE are collected upon packet reception. In IMU sensors, the accelerometer returns the acceleration force along x, y, and z axes including the gravity, and the gyroscope returns the speed of rotation around x, y, and z axes. Accelerometer and gyroscope readings are used to estimate the user's movement.

Data Processing Module. Raw RSSIs of Wi-Fi and BLE are transformed into RSSI stacking differences to represent the cumulative difference between an RSSI to other RSSIs [105]. Since RSSI contains the characteristics of each device, `PYLON` uses RSSI stacking differences to eliminate device dependency. It also uses them to generate virtual rooms for landmark correction.

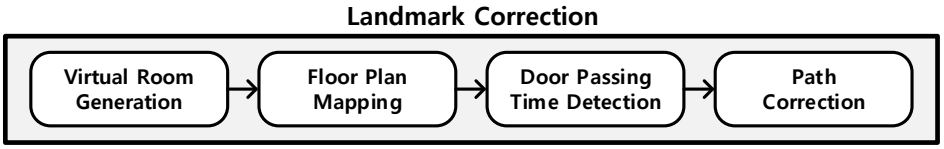


Figure 3.3: Detailed procedure of landmark correction. Landmark correction operates in four steps.

For IMU sensors, gyroscope data is integrated into the time domain to get the walking direction of the user, and passes through a low-pass filter to remove high-frequency noise. We count the peak of the acceleration magnitude of 3-axis to estimate the number of steps for the user, where each peak is considered one step. According to the interval of repetitive peaks, we determine the step frequency to estimate the step length. Then, we choose the step length estimation model that has a linear relationship with the step frequency. Combining the walking direction, step detection, and step length estimation, we roughly estimate the walking path of the user. Finally, the smartphone transmits RSSI stacking differences, movement and path information to the server.

3.3.3 Server Operation

Landmark Correction. The main role of landmark correction is to adjust the location and orientation of the user's path, using the real-world floor plan and collected data from the smartphone. In landmark correction, we use doors as landmarks, i.e., reference points, because the user should pass through a door to move from a room to another room. Figure 3.3 shows four components for landmark correction in detail that are essential in PYLON. First, applying a data mining approach for RSSI stacking differences, PYLON generates virtual rooms. Second, PYLON uses a simple and novel floor plan mapping algorithm to match virtual rooms to the real-world floor plan. Third, PYLON detects the exact time whenever the user passes through a door according to the floor plan mapping. We assume that PYLON knows the door location owing to the

real-world floor plan. Finally, if PYLON considers that the user goes through a door, it uses the location information of the door to correct the errors in the location and orientation of the user's path.

Particle Filter. The problem that the path may pass through walls after landmark correction still remains. Since walking through walls is not physically possible, we adopt a particle filter to adjust the estimated path so that the user's path does not touch the walls. The particle filter is a non-parametric form of Bayesian estimation, commonly used in computer vision and tracking. It helps to estimate the user's path and location with high accuracy.

3.4 Path Estimation

Step detection, step length estimation, and walking direction estimation are key elements to track the moving path of a user. Once PYLON is initialized, it continuously detects each step, and estimates walking direction and step length. The role of path estimation, i.e., dead reckoning, is to roughly generate the walking route of the user, without requiring very high accuracy. We apply landmark correction and then the particle filter for an accurate estimation of the user's path and location.

3.4.1 Step Detection

PYLON detects the user's step in accelerometer readings. Since the maximum magnitude of 3-axis of the accelerometer occurs when the user's heel strikes the ground [106], we design a step detection algorithm to detect the peak of accelerometer reading. To make step detection independent of the orientation of the smartphone, we consider only the magnitude of 3-axis [29]. We simply obtain the magnitude of the accelerometer reading as

$$a = \sqrt{(a_x^2 + a_y^2 + a_z^2)}. \tag{3.1}$$

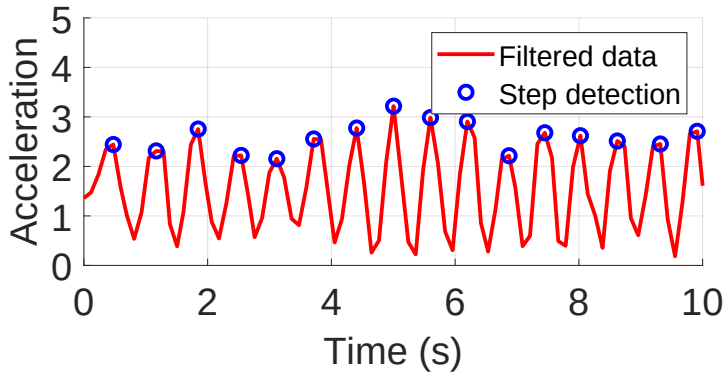


Figure 3.4: Peak detection using accelerometer readings.

To extract the smoothed magnitude of the accelerometer reading, we first use a low-pass filter that removes high-frequency noise, caused by random movement of the smartphone. The low-pass filter operates online on the smartphone, using the following exponentially weighted moving average:

$$a_{avg} = (1 - \alpha) \cdot a_i + \alpha \cdot a_{avg}, \tag{3.2}$$

where a_i is the ith raw magnitude of the accelerometer reading, and a_{avg} is its average after passing through the low-pass filter. The default value of α is set to 0.8.

After the low-pass filter smoothing, we apply a peak detection algorithm that uses a sliding window to find peaks of a_{avg}. If a_{avg} is larger than the other sample values collected within the window size t_w, and the slope product of the rightmost and leftmost samples of a_{avg} is less than 0 (i.e., slope sign change), we consider the sample of a_{avg} a peak (i.e., user's step). Since the user's step frequency is generally lower than 3Hz [35], the default value of t_w is set to 0.33. Figure 3.4 shows filtered accelerometer readings, where the red line represents filtered data and blue dots indicate detected peaks, i.e., steps. It shows our step detection algorithm performs well.

3.4.2 Step Length Estimation

There are several barriers for estimating the accurate step length for each user. It is not practical to ask each user to manually label step data so that the system trains the step model. Since body profiles such as height and weight, affect estimation accuracy, the generic model alone can not lead to high accuracy. In addition, the user's step model may change over time even for the same person. We apply the step model proposed in [35] to address the above issues. The authors in [35] propose a personalization algorithm that starts with a generic model and collects user data on the fly to train the model. They set up a system to collect more than 4000 steps from 23 users with various physical characteristics. They choose the frequency model [107] as their generic step model, which shows a clear trend that the step length has a linear relationship with the step frequency. The step frequency, i.e., walking frequency, is the number of detected steps per second. We express the model as

$$L_s = a \cdot f_s + b, \tag{3.3}$$

where L_s is the estimated step length, f_s is the step frequency, and a and b are the coefficients.

3.4.3 Walking Direction

Walking direction estimate is an important component in path estimation. For this, we use the gyroscope, which measures the angular velocities around x, y, and z in rad/s. The degree of rotation, i.e., walking direction, is obtained by integrating gyroscope readings. During a turn, the axis of rotation of the body always faces the center of the earth (i.e., the direction of gravity) [29]. The same axis as the direction of gravity shows the highest accelerometer reading on average. Since the gyroscope measures angular velocities on each axis of the smartphone, we first determine the orientation of the smartphone using the gravity value of 3-axis of the accelerometer.

PYLON uses the rotation degree of a single axis to represent the direction of gravity. While the rotation degree of the single axis can not perfectly measure the rotation of the user, PYLON applies additional algorithms of landmark correction in Section 3.6, to compensate for inaccuracy of estimation generated from the single axis. We also apply the low-pass filter to the rotation degree to eliminate high-frequency noise.

3.4.4 Location Update

We update the user's location according to step detection, step length estimation, and walking direction estimation. Basically, location update is performed per step basis. PYLON detects the user's step and calculates the location of the user based on the estimated step length and walking direction. All steps of the user are considered to be heading to the walking direction at the estimated step length. Since we have no prior knowledge of the initial location and orientation of the user, we estimate only the walking path. If the user takes the k^{th} step, we update the k^{th} location as

$$\begin{aligned} x_k &= x_{k-1} + L_s \cdot cos(\theta), \\ y_k &= y_{k-1} + L_s \cdot sin(\theta). \end{aligned} \tag{3.4}$$

As noted earlier, L_s is the estimated step length, θ is the walking direction, and x_k and y_k are the location of the smartphone from the perspective of the initial location. However, since we do not know the initial location and orientation of the user, we can not confirm the user's exact location as shown in Figure 3.5. Therefore, we apply landmark correction to address this issue. Again, path estimation roughly generates the walking path of the user, and PYLON corrects the location and orientation of the user with landmark correction.

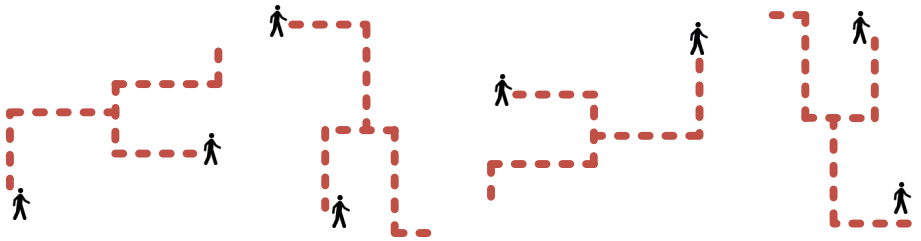


Figure 3.5: Examples of the possible walking path without additional information such as initial location and orientation.

3.5 Landmark Correction Part 1: Virtual Room Generation

We generate virtual rooms according to RSSI stacking differences, and conduct floor plan mapping, door passing time detection, and path correction. In this section, we first describe how to generate virtual rooms and a virtual floor plan. We use virtual rooms and a real-world floor plan to create a virtual floor plan and a physical floor plan, respectively, and perform floor plan mapping.

3.5.1 RSSI Stacking Difference

The first step for virtual room generation is collecting RSSI data of Wi-Fi and BLE on the smartphone. There is no human intervention when the smartphone collects RSSI data. The user just walks or sits in an office, shopping mall, or coffee shop. The smartphone collects RSSIs of Wi-Fi and BLE upon packet reception. If there are n Wi-Fi APs and BLE beacons in total in a building, we can represent RSSI samples of n Wi-Fi APs and BLE beacons as

$$R = [r_1, r_2, \cdots, r_n], \tag{3.5}$$

where r_i denotes the RSSI of the ith Wi-Fi AP or BLE beacon. We can use raw RSSIs as they are, but they are very different on various devices even if devices are at the same location and time [104]. Therefore, we use the difference relationship between RSSI values to overcome the device dependency problem. We leverage the concept of

RSSI stacking difference to represent the cumulative difference between one RSSI and other RSSIs [105]. RSSI stacking differences embody RSSI gap relations at specific locations and times, and show a relatively more stable feature of radio signals than raw RSSI values. We transform RSSI to the RSSI stacking difference, using

$$w_i = \sum_{j=1}^{n} I(r_i - r_j > 0)(r_i - r_j), \tag{3.6}$$

where I is an indicator function, i.e., characteristic function. The indicator function can be represented as

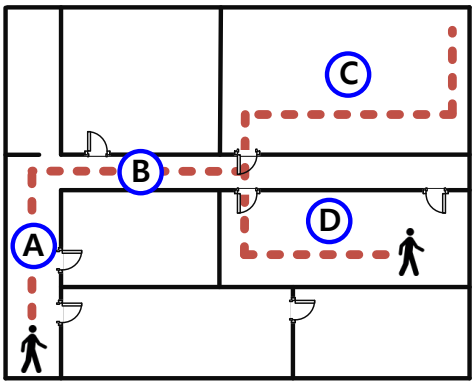
$$I(z > 0) = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{else,} \end{cases} \tag{3.7}$$

where all elements of z that are larger than 0 have the value 1. Based on the RSSI stacking difference, PYLON generates virtual rooms and a virtual graph.

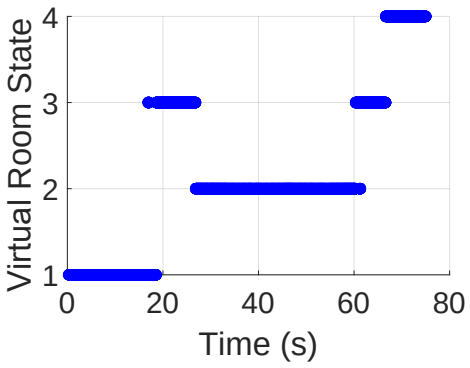
3.5.2 Virtual Room Generation

PYLON generates virtual rooms based on RSSI stacking differences and determines the number of rooms that the user passes through. To this end, we apply a data mining approach for RSSI stacking differences and adopt K-means clustering technique. K-means clustering demonstrates its high accuracy and efficiency in [105]. To apply K-means clustering to our problem, we should determine the cluster number k that equals the number of rooms that the user actually passes through. We choose the elbow method [108] to determine the cluster number. For easy understanding, we take an example in a real-world floor plan of a building as shown in Figure 3.6.

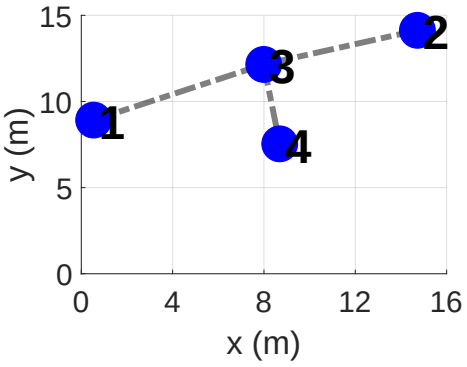
A user walks through corridors and rooms alphabetically, starting at corridor *A* and ending at room *D* as shown in Figure 3.6a. The smartphone collects RSSIs of Wi-Fi and BLE while the user is walking, and calculates RSSI stacking differences. The server determines the cluster number, using the elbow method, and conducts K-means clustering. In this example, the cluster number k is determined as 4, which equals the number of rooms that the user passes through. All data including RSSI stacking



(a) An example of user trace roaming 4 rooms and corridors



(b) A virtual room generation



(c) A virtual graph example

Figure 3.6: Example of a user trace in the real-world building and, virtual room generation, and a logical floor plan.

differences and information of path estimation is divided into four clusters. Figure 3.6b shows four virtual rooms generated by K-means clustering, and y axis represents the virtual room state. The state of each virtual room is randomly assigned by clustering. There are four state transitions between virtual rooms over time. In addition, we can infer the relationship between virtual rooms from Figure 3.6b. That is, the virtual room 3 is connected to all the other virtual rooms 1, 2, and 4.

3.5.3 Virtual Graph Generation

Based on virtual rooms and their relationships, we construct a logical floor plan of virtual rooms. The logical floor plan illustrates the relationship between virtual rooms, using a graph. The graph is formally defined as an undirected graph $G = (N, E)$ where a node $n \in N$ denotes a virtual room and an edge $(u, v) \in E$ represents that virtual rooms u and v are reachable from each other. The logical floor plan of virtual rooms is referred to as *virtual graph* G_v hereafter.

We generate G_v according to the relationship between virtual rooms as shown in Figure 3.6b. The user moves from virtual room 1 to 3 at 20 s first. According to the relationship, we generate two nodes, i.e., nodes 1 and 3, and one edge, i.e., $(1, 3)$, as part of G_v. The virtual room state changes from virtual room 3 to 2 at 24 s. Then we add a new node 2 and an edge $(2, 3)$ to G_v. The virtual room state changes from room 2 to 3 at 62 s. However, since virtual rooms 2 and 3 are already in G_v, there is no change in G_v. Finally, we add a new node to G_v at 68 s. The virtual room state changes from state 3 to 4. Then we add node 4 and edge $(3, 4)$ to G_v. Figure 3.6c represents the complete G_v, showing reachability between nodes. We assume that the reachability is undirected. This means, if node 3 is reachable from node 1, then node 1 is reachable from 3. Therefore, node 3 is reachable from all the other nodes according to G_v.

3.5.4 Physical Graph Generation

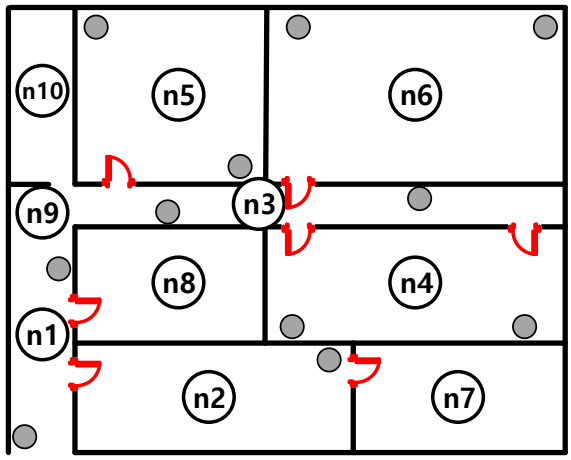
After constructing virtual graph G_v, we should map G_v to a real-world floor plan. Before mapping, we also need a logical graph of the real-world floor plan. For convenience, the logical floor plan of the real-world floor plan is referred to as *physical graph* G_p. G_p is also modeled as an undirected graph. We consider each room and corridor a node in G_p, and divide a corridor into several segments as shown in Figure 3.7a. We first divide a corridor into vertical and horizontal segments, for instance, $n1$ and $n3$. The area of a corridor overlapped with vertical and horizontal segments is considered an independent node, such as $n9$, because it embodies RSSI characteristics of both $n1$ and $n3$. Therefore, when generating virtual rooms, $n9$ may be considered an independent area depending on the clustering result. Considering the reachability between corridors and rooms, we generate G_p based on the real-world floor plan as shown in Figure 3.7b.

3.6 Landmark Correction Part 2: From Floor Plan Mapping to Path Correction

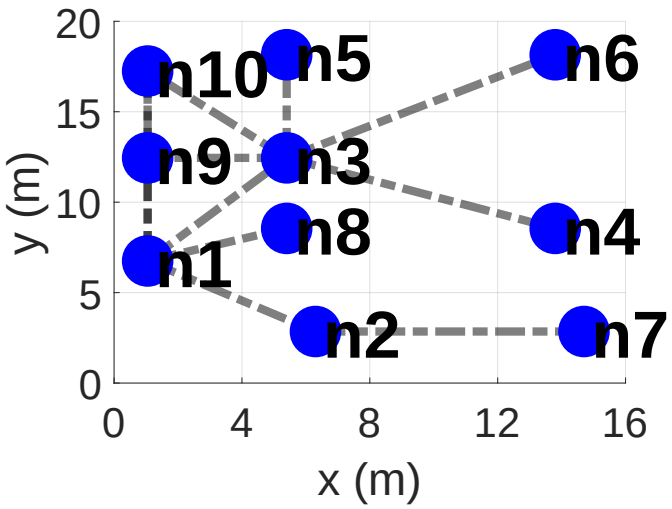
After constructing a virtual graph G_v and a physical graph G_p, we now conduct floor map mapping between G_v to G_p in two steps: backbone node mapping and dead-end node mapping. Before doing this, we determine *candidate graph* G_c first. G_c is a subset of G_p and consists of the same number of nodes as G_v. We regard a subset graph as G_c for G_v that all nodes composing G_c are *reachable*.

3.6.1 Candidate Graph Generation

Basically, the number of nodes in G_v is less than that in G_p, unless the user moves around all the rooms of the real-world floor plan. Therefore, more than one subset of G_p is a candidate for G_v. If we denote the number of nodes in G_p and G_v as N_p and N_v, respectively, the maximum number of candidates for G_c is given by N_v combination



(a) A real-world floor plan of the building and the location of Wi-Fi APs and BLE beacons with gray dots



(b) A physical graph

Figure 3.7: A real-world floor plan and a physical floor plan. All rooms and corridors are considered nodes.

of N_p. However, all subsets can not be G_c. If one subset contains a node that is not reachable from the other nodes, this subset should be removed from G_c. Since all rooms are physically connected, all nodes should be reachable from each other.

We give some examples for subsets of G_p. In the case of two subset graphs in Figure 3.8a, all nodes composing each subset graph are reachable from each other. Therefore, these two subset graphs can be G_c. However, in Figure 3.8b, two subset graphs can not be G_c because $n2$ in the left figure, $n4$ and $n7$ in the right figure lose their reachability from the other nodes. We exclude these subset graphs from G_c candidates. Therefore, if G_v contains four nodes, 39 subset graphs can be selected as G_c from 210 subset graphs in the case of G_p shown in Figure 3.7b. The goal of floor plan mapping is to choose only one G_c as G_v.

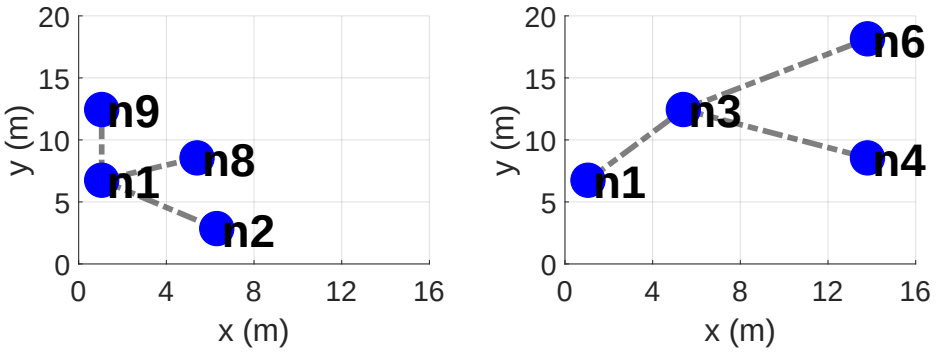
3.6.2 Backbone Node Mapping

Now, we conduct floor plan mapping according to G_c. The first step is backbone node mapping. In graph theory, *betweenness centrality* is a measure of the node's centrality [109]. Betweenness centrality of each node is measured as the number of the shortest paths that pass through that node. A node that embodies many shortest paths between other nodes has high betweenness centrality. Generally, betweenness centrality of an leaf node in the graph is 0. We can express betweenness centrality of node v as

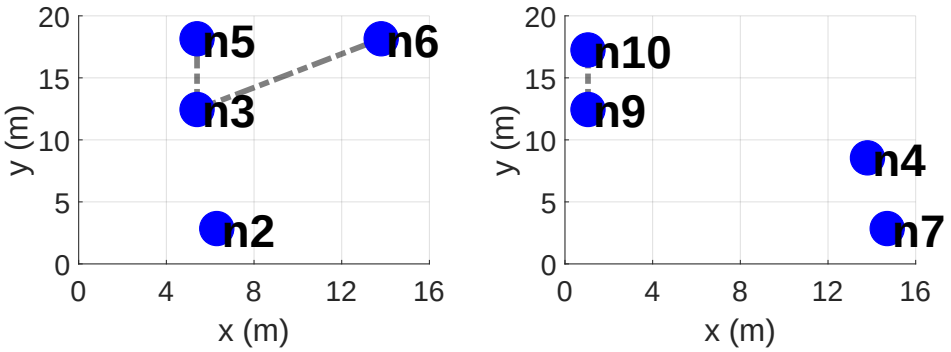
$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}, \tag{3.8}$$

where σ_{st} is the total number of the shortest paths from node s to t, and $\sigma_{st}(v)$ is the number of the paths that pass through v. In Figure 3.7b, $n1$ and $n3$ apparently have higher betweenness centrality than the other nodes, and nodes that have non-zero betweenness centrality are called *backbone nodes*. We first perform backbone node mapping between G_v and G_c, in order of high betweenness centrality.

Figure 3.9 shows the example results of backbone node mapping between two G_c's. The left and right graphs represent G_c and G_v, respectively, in Figs. 3.9a and 3.9b.

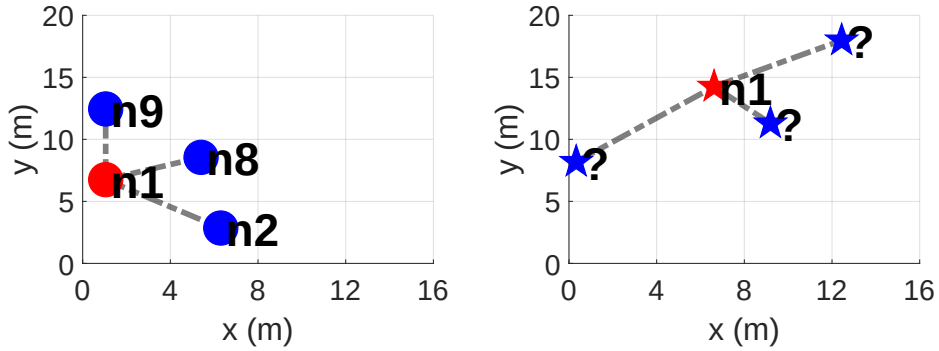


(a) The subset graphs that can be candidates

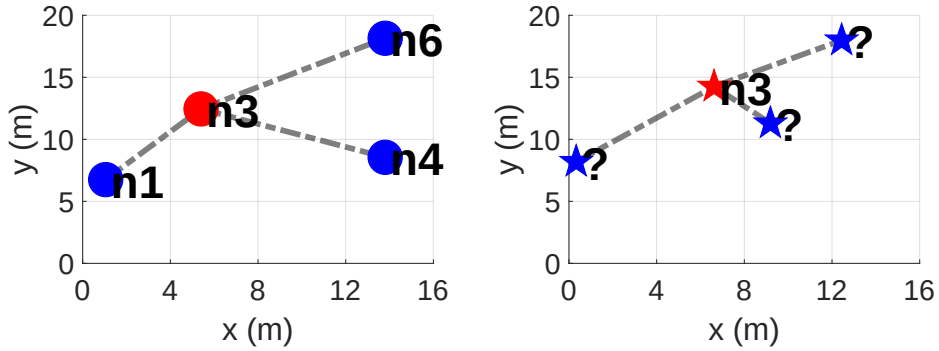


(b) The subset graphs that lost reachability

Figure 3.8: Subset graphs of the physical graph.



(a) The first candidate graph and the virtual graph



(b) The second candidate graph and the virtual graph

Figure 3.9: Two examples of candidate graphs and virtual graphs after backbone node mapping.

Nodes for G_c and G_v are illustrated with circles and stars, respectively. We represent nodes mapped by backbone node mapping in red color. Since betweenness centrality of $n1$ and $n3$ is 3 in G_c, they are mapped first because their betweenness centrality is the highest. Since leaf nodes have betweenness centrality of 0, they are not backbone nodes. We perform backbone node mapping in all 39 G_c's, and then conduct dead-end node mapping to match the rest of the nodes.

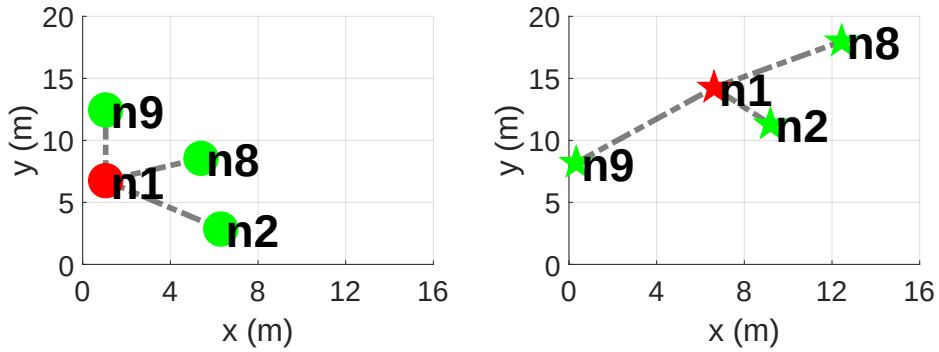
3.6.3 Dead-end Node Mapping

Dead-end nodes represent the rest of nodes that are not mapped by backbone node mapping. Generally, dead-end nodes are leaf nodes of a graph, and logically symmetric in most cases. Mapping for dead-end nodes is not an easy task, relying on only the logical graph G_v and G_c. Therefore, we use a relative location of each node to map dead-end nodes perfectly. In G_c, we can determine the exact location of each node according to the real-world floor plan. We define the location of each node as the center of a room or corridor, following the real-world floor plan. When PYLON generates G_v using RSSI stacking differences, we can obtain the relative location of each node in G_v, thanks to path estimation. Therefore, we can determine the relative location of each node in both G_v and G_c. Leveraging the relative locations, we conduct dead-end node mapping.

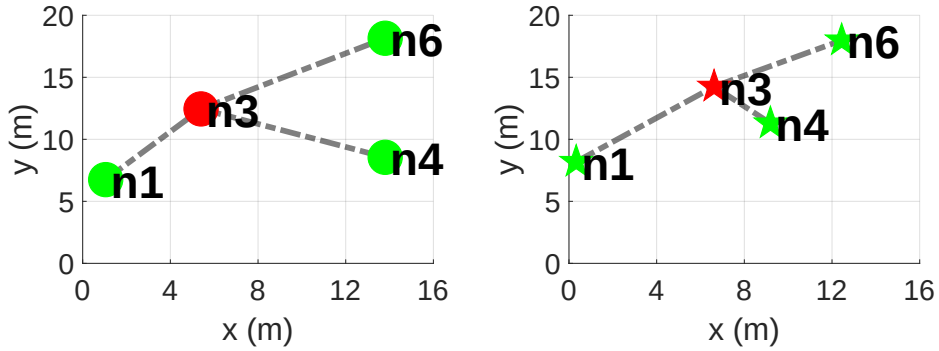
Cosine similarity is a measure for similarity between two non-zero vectors of an inner product space, which measures the cosine of the angle between them [110]. It is not a magnitude but the judgment of orientation. The cosine similarity has a value between -1 and 1. If two vectors have the same orientation, the cosine similarity is 1. Two vectors in diametrically opposite orientation have the cosine similarity of -1, independent of their magnitude. We can represent the cosine similarity of two vectors A and B as

$$cosine\ similarity = cos(\theta) = \frac{A \cdot B}{||A||||B||}. \tag{3.9}$$

We leverage the concept of cosine similarity for dead-end node mapping, using backbone nodes. For easy understanding, we take two examples for dead-end node mapping as shown in Figure 3.10. Nodes mapped by dead-end node mapping are colored in green. In the case of Figure 3.10a, we generate three vectors from $n1$ to $n2$, $n8$, and $n9$ in each G_c and G_v. Therefore, we generate a total of 9 $(= 3 \times 3)$ cosine similarity values. PYLON first conducts node mapping in order of high cosine similarity. In this example, since the vectors from $n1$ to $n8$ in G_c and G_v show the highest cosine similarity, $n8$ is mapped first. Afterwards, $n2$ is mapped following the above



(a) The first candidate graph



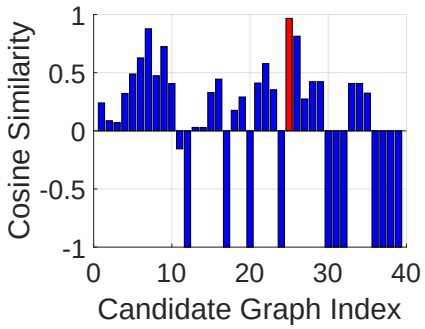
(b) The second candidate graph

Figure 3.10: Two examples of candidate graphs and virtual graph after dead-end mapping.

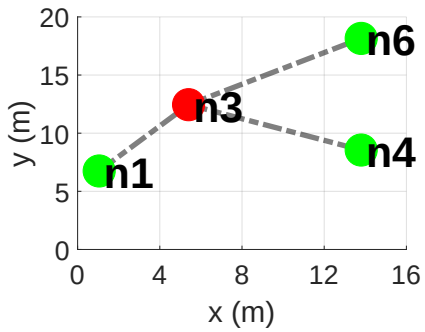
process, and $n9$ is mapped last. In this way, we conduct dead-end node mapping until all the nodes are mapped. In the case of Figure 3.10b, we generate three vectors from $n3$ to $n1$, $n4$, and $n6$ in each G_c and G_v. According to the cosine similarity, $n4$, $n6$, and $n1$ are mapped in order.

3.6.4 Final Candidate Graph Selection

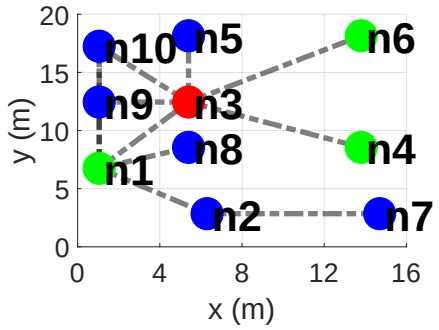
If one G_c exactly fits G_v, the cosine similarity value of each vector from backbone nodes to all dead-end nodes should be 1 in both G_c and G_v. Therefore, we choose one G_c as G_v, which shows the highest average cosine similarity. Figure 3.11a shows the



(a) The average cosine similarity of 39 candidate graphs



(b) 25^{th} candidate graph that shows the highest average cosine similarity



(c) The physical graph completing the mapping algorithm

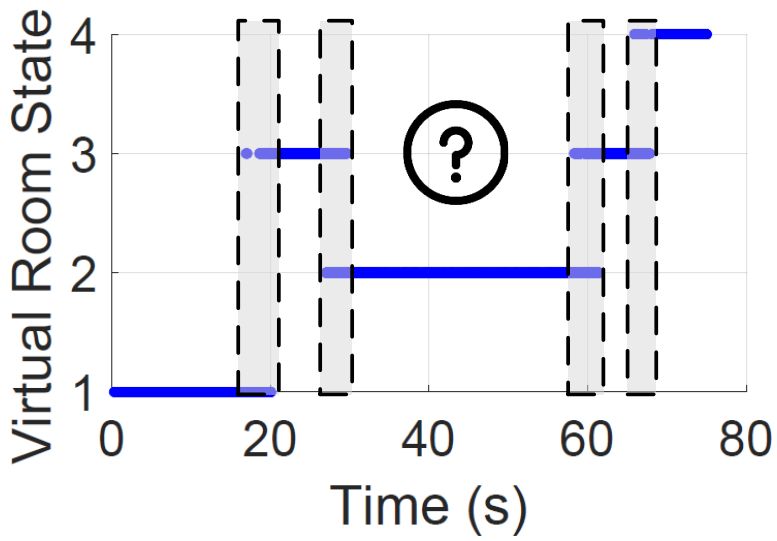
Figure 3.11: The average cosine similarity of candidate graphs and final answer of mapping algorithm.

average cosine similarities for all the 39 G_c's. Among 39 G_c's, the 25^{th} G_c shows the highest cosine similarity, which we choose as G_v. Figure 3.11b shows that the 25^{th} G_c is composed of $n1$, $n3$, $n4$, and $n6$, and Figure 3.11c shows the exact position of the 25^{th} G_c in the real-world floor plan. From the mapping results, we know about transitions between rooms or corridors in the real-world floor plan over time. Therefore, we can correct the location and orientation of the user's path, using landmarks.

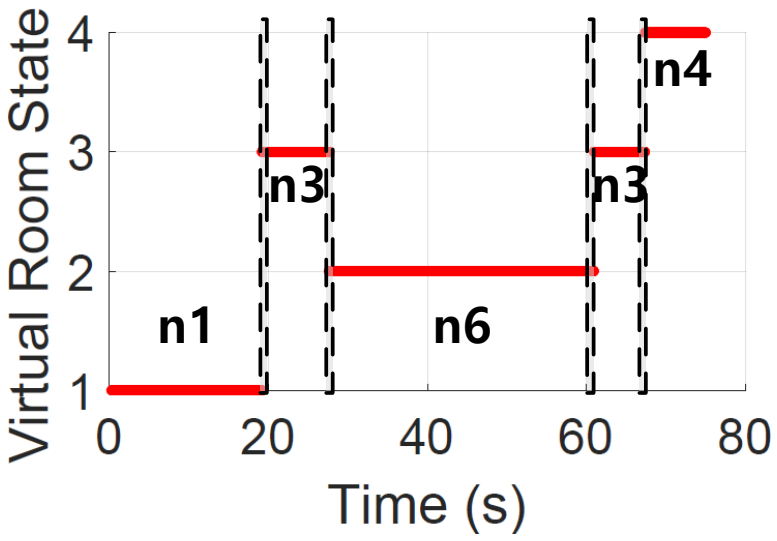
3.6.5 Door Passing Time Detection

Backbone node mapping and dead-end node mapping aim to correct the location and orientation of the user's path, using landmarks. We leverage room entrances, i.e., doors represented on the real-world floor plan, as landmarks to compensate for the cumulative error of IMU sensor readings. Since the real-world floor plan of a building is given, door locations are known values. We use the location of each door to compensate for the cumulative error and should find the exact time when the user passes the door. If we know the exact door passing time, we can correct the location and orientation of the user's path according to the door location. Therefore, we reuse a clustering result as shown in Figure 3.6b to find the exact door passing time. The clustering result shows room state changes over time. The more accurate the door passing time, the better the accuracy in path estimation and localization.

However, it is challenging to find the exact door passing time, using the clustering results alone. The time period during which a virtual room state undergoes the ping-pong effect is observed when the room state changes as shown in Figure 3.12a. This means that the clustering result can not directly imply the exact door passing time. To solve this problem, we design a sliding window to minimize the ping-pong effect and to prevent the distortion of state change. We consider at least two steps are required to completely change the room state that the user stays in. The sliding window helps to determine how many beacons we can receive while changing two steps. We can



(a) The old virtual room state that suffers from the ping-pong effect



(b) The new virtual room state after applying the sliding window

Figure 3.12: The clustering results before and after applying the sliding window.

express the window size as

$$W = \frac{2}{f_s} \cdot \frac{1}{I_{beacon}} \cdot S_N, \tag{3.10}$$

where f_s is the step frequency, I_{beacon} is the beacon interval, and S_N is the number of Wi-Fi APs or BLE beacons that the smartphone can receive. Since RSSI samples are collected whenever the smartphone receives beacons when PYLON generates virtual rooms, we set the unit of the sliding window to the number of beacons. We apply the sliding window to the old state (i.e., the result of clustering) to generate a new state, which eliminates the ping-pong effect, as follows.

$$state_{new}(i+W) = mode(state_{old}(i+1, i+2, \cdots, i+W)), \tag{3.11}$$

where i represents the i^{th} state. We take a mode value (i.e., the state that appears most often) of the old state in the sliding window as the new state. As a result, we successfully eliminate the ping-pong effect, and find the exact door passing time when the state changes, as shown in Figure 3.12b.

3.6.6 Path Correction

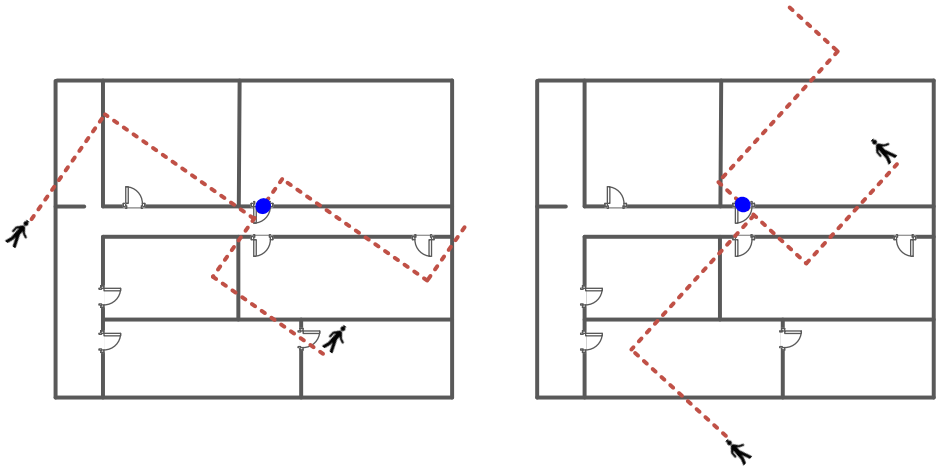
We are now ready to correct the user's path and location, using the door passing time. Note again that PYLON maps virtual rooms to the real-world floor plan through virtual room generation and floor plan mapping, and finds the exact door passing time based on the clustering result. If the user passes a door, PYLON determines the exact location and orientation of the user's path. For the sake of clarity, we use an example to explain the path correction. First, we should know when the user passes the first door. After conducting floor plan mapping, we know about room state changes of the path in the real-world floor plan as shown in Figure 3.12b. The room state change from $n3$ to $n6$ at $24\ s$ is important. Since $n3$ is a corridor and $n6$ is a room, the user goes through the door for the first time at $24\ s$. Therefore, using parallel transference, we move the user's path to the location of the first detected door.

We mark the location of the first door with a blue dot in Figure 3.13, and move the path up to the blue dot, used as an anchor. However, we still can not have confidence about the orientation of the user's path as shown in Figure 3.13a. To find its orientation, we take an assumption that the user moves vertically into the door when passing. Therefore, PYLON modifies its orientation to the vertical angle of the door, and finds the exact path as shown in Figure 3.13b. Each time the user passes through a door, PYLON corrects the location and orientation of the user's path.

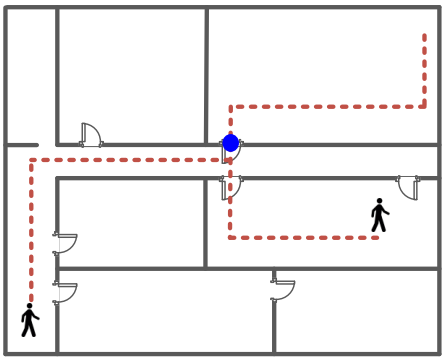
3.7 Particle Filter

PYLON realizes path estimation without human intervention, thanks to landmark correction. However, the problem that the user passes through a wall still remains. Figure 3.14 shows the path estimation by landmark correction, which is not perfect. The black dotted line represents the ground truth path of the user, the gray line, i.e., *Legacy*, shows the estimated path through dead reckoning, which leverages IMU sensors only, and the blue line is the estimated path after landmark correction (*LC*). For *Legacy*, the initial location and orientation of the smartphone is provided for easy comparison. Compared to *Legacy*, *LC* provides higher accuracy in path estimation after the user passes through the first door. However, *LC* shows that the user still passes through walls in the area marked with two circles in yellow. Since this is not physically possible, we apply a particle filter to keep the user's path in a corridor or room with the help of the real-world floor plan [35].

To keep the user's path in a corridor or room, we adopt the particle filter whose concept is illustrated in Figure 3.15. Initially, the particles (i.e., samples) are generated around the estimated point, and all particles are equally weighted. Each particle moves according to the movement of the user step by step, and is constrained by the real-



(a) The parallel transference of the path to the first door



(b) The path corrected the orientation with door

Figure 3.13: The example of parallel transference and the orientation corrected path.

world floor plan. Therefore, the k^{th} location of the i-th particle is updated as:

$$\begin{aligned} x_k^i &= x_{k-1}^i + (L_s + \delta_k^i) \cdot cos(\theta + \gamma_k^i), \\ y_k^i &= y_{k-1}^i + (L_s + \delta_k^i) \cdot sin(\theta + \gamma_k^i), \end{aligned} \tag{3.12}$$

where L_s and θ are the estimated step length and the walking direction, respectively, while δ_k^i and γ_k^i are the zero mean Gaussian noise.

PYLON checks whether the movement of each particle violates any indoor constraints such as moving into the wall. These particles are considered dead, and we set their weight to 0. The remaining particles (i.e., live particles) equally absorb the total weight of dead particles, which increases the weight of live particles. Then, through the re-sampling process, we generate new particles around the live particles, and assign the weight of the dead particles equally to the new ones. This makes the total weight of all the particles greater than 1. Therefore, we normalize the weight of all the particles by the total weight.

The distribution of particles reflects the likelihood of the real position. We calculate the weighted average of all the particles as the filtered point. In PYLON, we use 1000 particles to filter the estimated point of the user, and vary the number of particles from 100 to 5000. The localization accuracy does not increase when the number of particles exceeds 1000. Therefore, we set the number of particles to 1000, considering the localization accuracy and computational complexity. By doing so, the particle filter in PYLON helps the user's path correctly follow the ground truth value without passing through walls, different from that in *Legacy* and *LC*.

3.8 Performance Evaluation

3.8.1 Implementation and Measurement Setup

We conduct experiments for a test area of 418 (19×22) m^2 in an office building, and use four Wi-Fi APs and seven BLE beacons. In Figure 3.7a, doors are colored in red and Wi-Fi APs and BLE beacons are represented by small grey circles. Wi-Fi APs

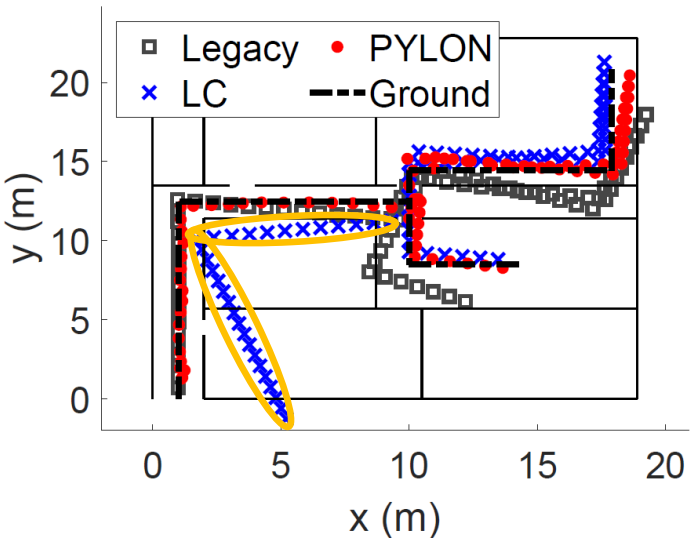


Figure 3.14: The paths of Legacy, LC, and PYLON.

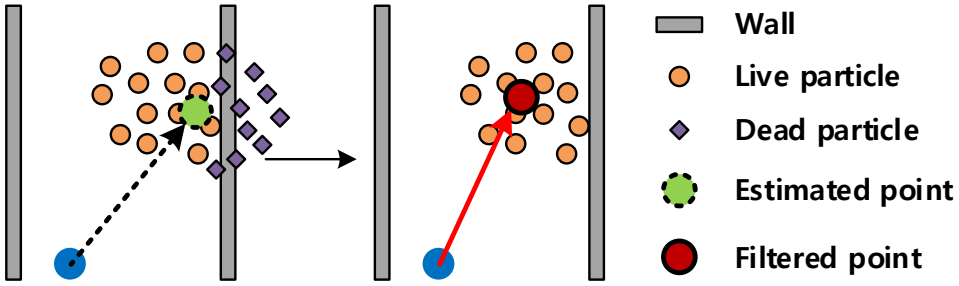


Figure 3.15: An illustrative example of the particle filter.

are commercial off-the-shelf devices, and BLE devices are Estimote BLE beacons that support iBeacon and Eddystone [111]. Each BLE beacon periodically transmits BLE advertising packets every 100 ms that equals the Wi-Fi AP beacon interval. We develop PYLON as an application on the Android OS of five smartphones to cover device heterogeneity: Nexus 5, Nexus 6p, Pixel 2, Pixel 3, and Galaxy S10.

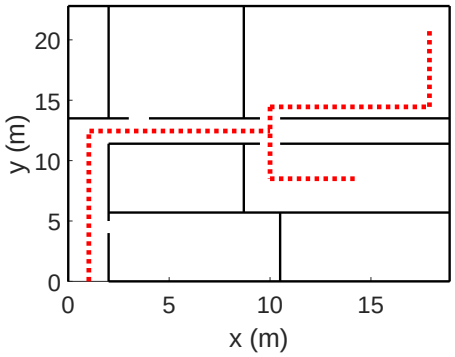
Since Nexus 5 was released in 2013 and Galaxy S10 in 2019, we experimented with the smartphones that have been released in the last 7 years. The application is plug-and-play and collects RSSI data of Wi-Fi/BLE, gyroscope, and accelerometer. Three users participate in performance evaluation of floor plan mapping accuracy, door passing time estimation, and localization including step detection and walking direction estimation. Each user holds a smartphone in hand. However, we did not impose any constraints on the direction of the smartphone (e.g., horizontal or vertical holding) while walking. To evaluate the performance of path estimation and localization, we experiment with three different traces as shown in Figure 3.16.

We compare PYLON with the following three schemes: *Legacy*, *PF* and *LC*.

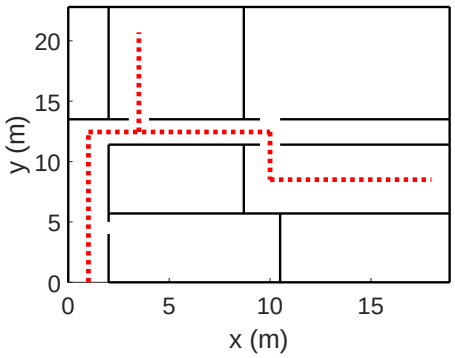
- *Legacy* estimates the user's location using only IMU sensors of the smartphone. It requires the initial location and orientation of the user. The result of path estimation module of PYLON is the same as that of *Legacy*.

- *PF* applies the particle filter to *Legacy*. It requires the initial location and orientation of the user to apply the particle filter [35].

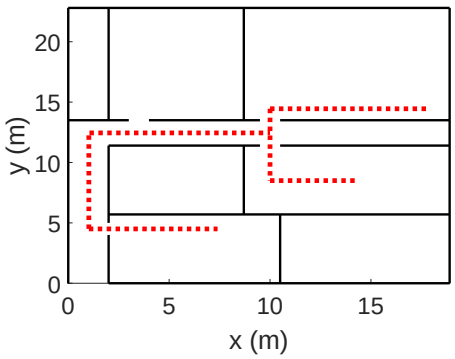
- *LC* applies only the landmark correction module to *Legacy*. It is a downgraded version of PYLON and does not require any information from the user like PYLON. Landmark correction compensates for location and orientation errors according to door locations.



(a) The first trace passing $n1$, $n3$, $n4$, and $n6$



(b) The second trace passing $n1$, $n3$, $n4$, and $n5$



(c) The third trace passing $n1$, $n2$, $n3$, $n4$, and $n6$

Figure 3.16: The example of three traces.

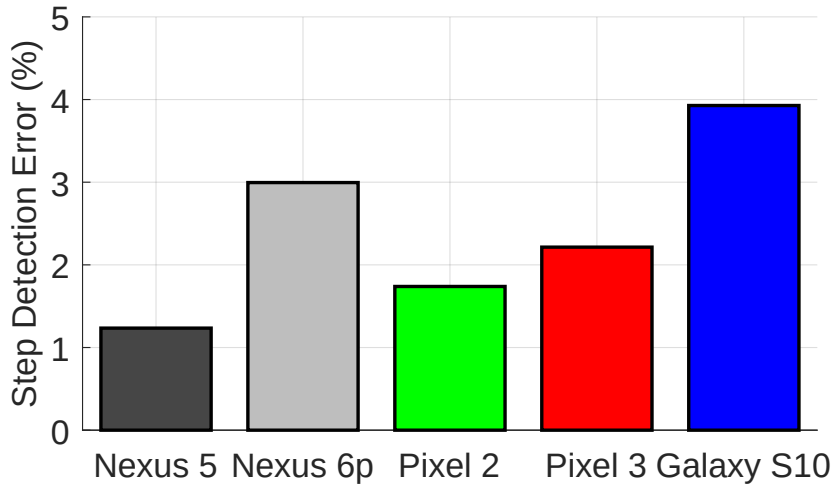


Figure 3.17: Performance of step detection.

3.8.2 Step Detection Accuracy

We first evaluate the performance of step detection on the five smartphones. In the case of the step detection, we conduct the experiment with 20 users. The average step detection errors are shown in Figure 3.17. Since *LC*, *PF*, and PYLON correct the location and walking direction error in their own way, the step detection is the only common part of PYLON and all the comparison schemes. The errors in step detection are below 3.2% in all the smartphones, and the average error on the five smartphones is 2.2%, which means a step counting error of 2.2 in 100 steps. Landmark correction of PYLON contributes to the compensation for these small errors.

3.8.3 Floor Plan Mapping Accuracy

As shown in Figure 3.16, traces 1 and 2 contain four rooms and two doors, and trace 3 contains five rooms and three doors. The accuracy of floor plan mapping affects the overall performance of PYLON. We first show the snapshots of the ground truth floor plan mapping of the three traces. Figure 3.18 shows the graphs of virtual rooms after

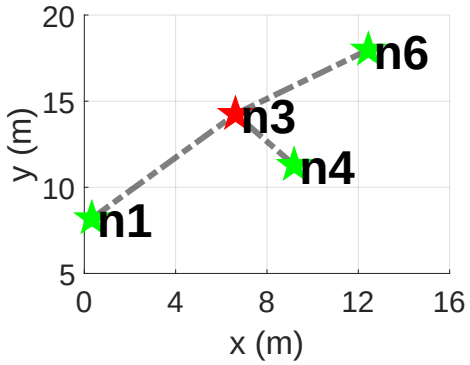
conducting floor plan mapping. Red and green colored stars represent backbone nodes and dead-end nodes, respectively. According to the ground truth floor plan mapping results, we measure the mapping accuracy of all the smartphones with respect to the three traces.

Note again that the average cosine similarity for floor plan mapping represents the similarity between the final candidate graph and the virtual graph. Figure 3.19 and Figure 3.20 show floor plan mapping accuracy and the average cosine similarity of the three traces. In the both figures, the x-axis represents the trace number. Figure 3.19 shows that all the schemes show higher than 90% of floor plan mapping accuracy regardless of the smartphones and traces. The average mapping accuracy of all the smartphones and traces reaches 97%, which is sufficiently high. Figure 3.20 shows the average cosine similarity between the candidate graph and the virtual graph after conducting mapping. The average cosine similarity is higher than 0.91 for each smartphone and trace. The average cosine similarity of all the smartphones and traces is 0.96, which shows that the floor plan mapping is successful, leading to the high similarity between G_c and G_v.

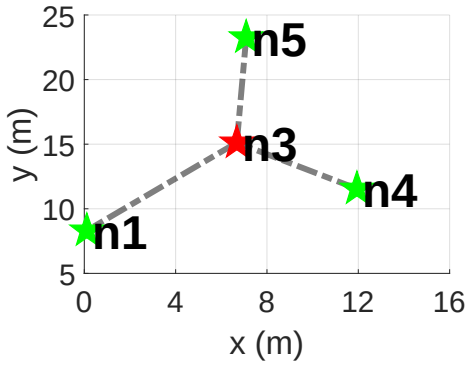
3.8.4 Door Passing Time

After successfully conducting floor plan mapping, PYLON determines the time when the user passes a door. Since landmark correction corrects the location and orientation of the user's path using the door passing times, finding the exact passing times affects the performance of path estimation and localization. When the difference between the ground truth time and the estimated time by PYLON is large, landmark correction rather increases the inaccuracy of the path estimation and localization. To collect the ground truth data, we record the time through PYLON application whenever the user passes a door.

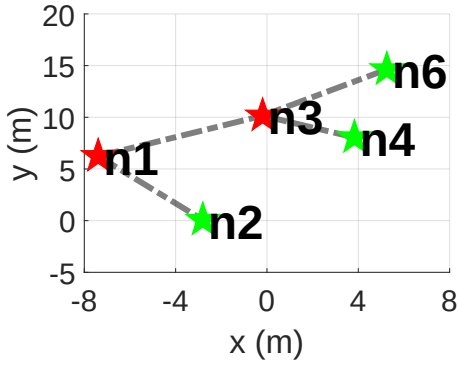
Figure 3.21 shows the CDF of the difference between the ground truth time and the estimated time in PYLON. The results show that the time differences are smaller



(a) Trace 1



(b) Trace 2



(c) Trace 3

Figure 3.18: The examples of three traces for correct floor plan mapping.

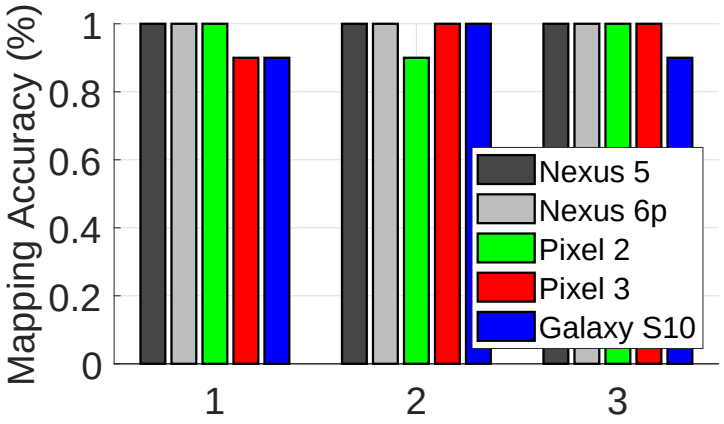


Figure 3.19: Accuracy performance of floor plan mapping.

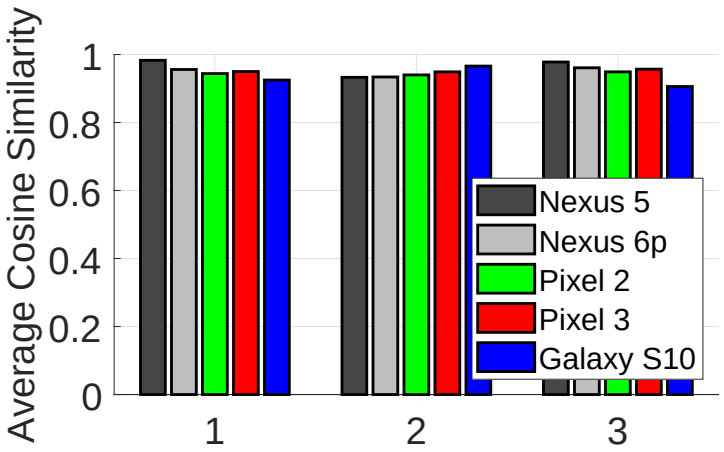


Figure 3.20: The average cosine similarity of mapping result.

than 2 s and the average differences lie between 0.57 and 0.72 s for all the smartphones. Considering that the average step frequency of the users is 0.69/s, PYLON detects the door passing time in one step on average. Considering that at least two steps are required for room change, i.e., passing the door, PYLON accurately detects the door passing time. Figure 3.22 shows a snapshot of walking direction, i.e., orientation, corrected by landmark correction according to the door passing time. Landmark correction compensates for walking direction errors whenever the user passes doors at 24, 40, and 60 s, and consequently the walking direction errors remain below 10 degrees in *LC*. However, in *Legacy*, gyroscope readings contribute to error accumulation, therefore the walking direction error increases up to 27 degree for 75 s.

3.8.5 Walking Direction and Localization Performance

We evaluate path estimation performance of PYLON in terms of errors in walking direction estimation and localization. We record the ground truth locations of the user during the experiments to obtain the walking direction estimation and localization performance. PYLON compares the direction and distance difference of each point between the ground truth location and the estimated location. Figure 3.23 shows the walking direction estimation performance of PYLON and the comparison schemes. Figure 3.23a shows the CDF of walking direction estimation errors for Nexus 5, and Figure 3.23b presents average performance on the remaining smartphones with bar graphs. In Figure 3.23a, PYLON shows the highest walking direction estimation performance compared with the other schemes.

Legacy does not correct the user's orientation, so it shows the worst performance in the walking direction estimation. *PF* slightly reduces the walking direction estimation errors thanks to the particle filer. However, the particle filter can not correct the orientation error greatly in the case of a room where particles do not hit a wall (i.e., staying alive). In terms of the walking direction estimation, *LC* outperforms *Legacy* and *PF* because the orientation is exactly corrected using the door passing times. Figure 3.23b

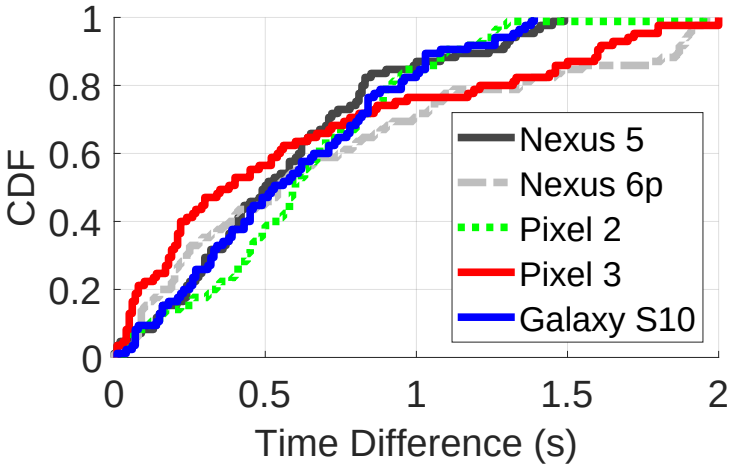


Figure 3.21: Average time difference of door passing time detection.

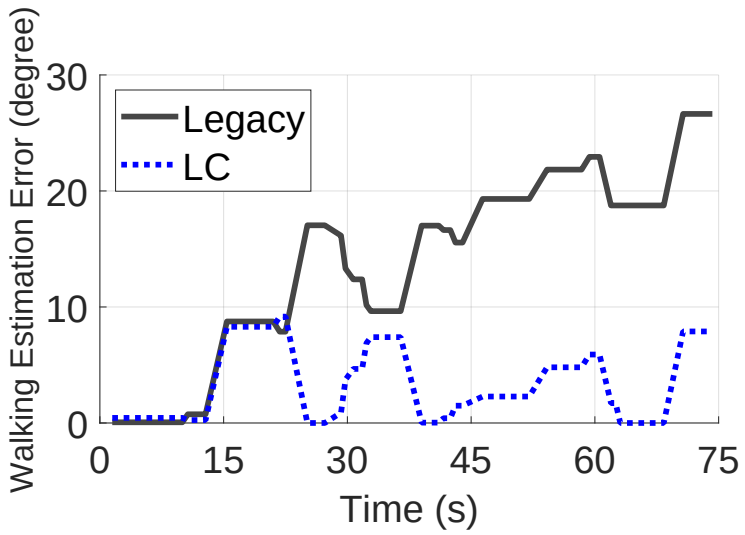
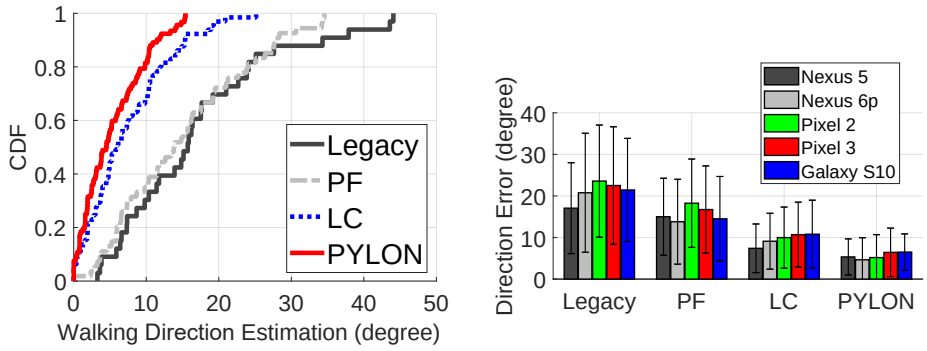


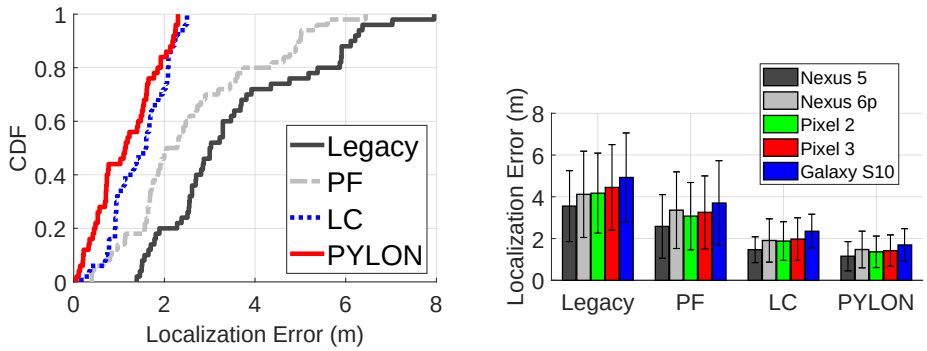
Figure 3.22: A snapshot of corrected walking direction in *LC*.



(a) The CDF of walking direction estimation error for Nexus 5

(b) Average walking direction estimation error

Figure 3.23: Walking direction estimation performance.



(a) The CDF of localization error for Nexus 5

(b) The average localization error

Figure 3.24: Localization performance.

shows the average walking direction estimation errors of all the schemes. The average walking direction estimation error also shows the same trend as mentioned before. PYLON shows the best performance regardless of smartphone types and its errors are below 6.4 degrees in all the smartphones. PYLON reduces the error by 75.2, 66.7, and 45.8% on average, compared to *Legacy*, *LC*, and *PF*, respectively.

Figure 3.24 shows localization performance in the same way as walking direction estimation performance. We first show the CDF results of Nexus 5 in Figure 3.24a. *Legacy* shows the worst localization performance with an error of up to 8 m, com-

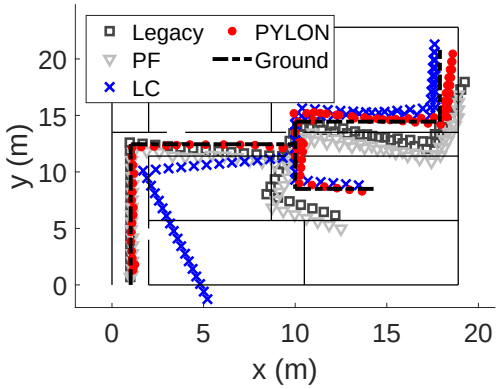
pared to the other schemes. *PF* shows better localization accuracy compared to *Legacy* thanks to the particle filter. PYLON shows the best localization performance with an error of less than 2.1 m, compared to the other schemes. Figure 3.24b depicts the average localization error which shows the same trend as the CDF results. PYLON achieves the highest localization accuracy. PYLON shows all the average errors less than 1.69 m, and reduces the localization error by 66.5, 55.4, and 26.1% on average, compared to *Legacy*, *LC*, and *PF*, respectively. The overall average error of PYLON regardless of device types is 1.42 m. We evaluate the walking direction estimation and localization performance of PYLON on the five smartphones, and demonstrate that PYLON outperforms the comparison schemes. PYLON minimizes cumulative IMU sensor reading errors using landmark correction, and does not allow the user's path to go through walls, thanks to the particle filter.

3.8.6 Impact of WiFi AP and BLE Beacon Number

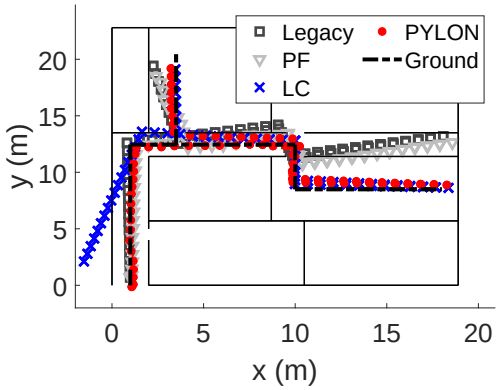
We initially leverage 11 WiFi APs and BLE beacons in the office building of size 418 m^2. Now we randomly eliminate WiFi APs and BLE beacons one by one to further examine the localization performance according to the number of WiFi APs and BLE beacons. As shown in Fig. 3.26, the localization error does not change until the number of WiFi APs and BLE beacons removed reaches 3. However, the error increases sharply from 5.38 to 12.68 when the number of WiFi APs and BLE beacons removed has increased from 4 to 9. For the removed number of greater than 4, since the floor plan mapping and door passing time are less accurate, the landmark correction can not efficiently correct the user's location. Therefore, at least 8 WiFi APs and BLE beacons are required for PYLON to work properly.

3.8.7 Impact of Walking Distance and Speed

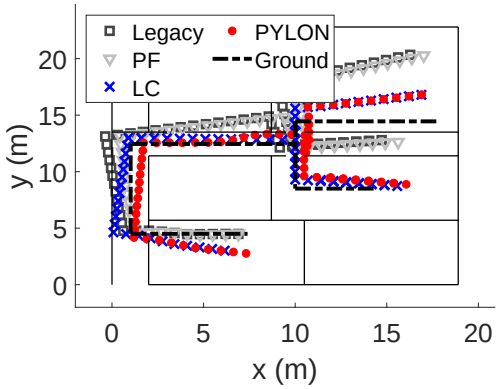
We change the walking distance from 50 m to 500 m to evaluate its effect on the localization error. In general, the drift of IMU sensors and localization error increase with



(a) Trace 1



(b) Trace 2



(c) Trace 3

Figure 3.25: The snapshots of path estimation for the three traces on Nexus 5.

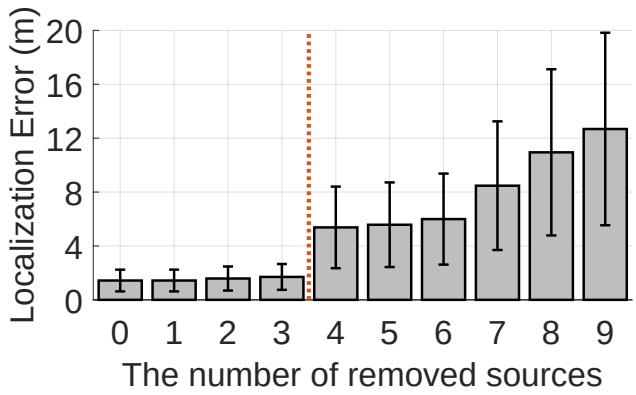


Figure 3.26: Localization error according to the number of WiFi APs and BLE beacons.

Table 3.1: Experimental details for different areas

Areas	Size (m^2)	#Sources	#Rooms	#Doors	Length (m)
1	$1,454$	17	5	3	58
2	$4,110$	39	18	9	270

the walking distance. In Fig. 3.27a, the localization errors stay around 1.38 and 2.02 m in the case of PYLON and *LC*, respectively. These schemes correct the user's location with landmark correction and can effectively handle drift accumulation in IMU sensors. However, the errors in *Legacy* and *PF* accumulate up to 27.20 and 7.66 m, respectively, with the walking distance. Even if *PF* corrects the user's walking path using the particle filter, it is not enough to solve the basic IMU sensor drift problem. We examine the performance of PYLON according to three different walking speeds: (1) slow (0.5 m/s), (2) normal (0.7 m/s) (3) fast (1 m/s). As shown in Fig. 3.27b, PYLON achieves an average accuracy of 1.19, 1.31, and 1.69 m, respectively, for each walking speed. The faster the user walks, the greater the movement of the human body. Therefore, the localization error increases due to fluctuations in IMU sensor readings, especially gyroscope readings.

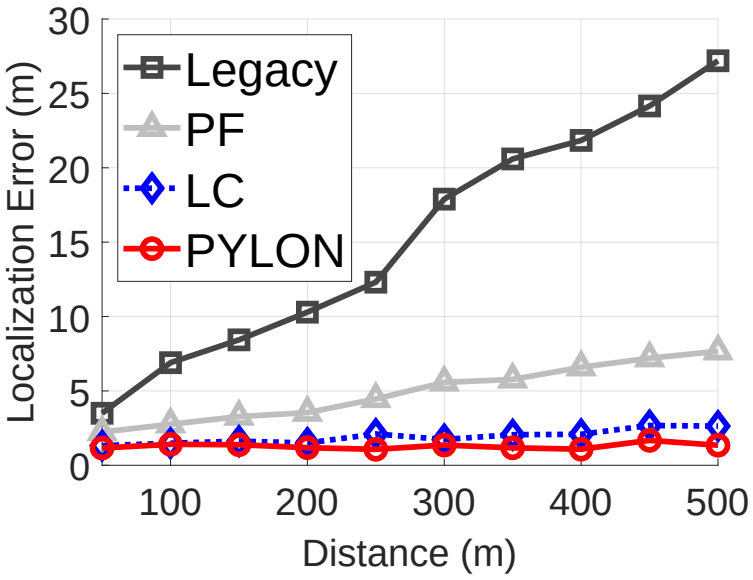
3.8.8 Performance on Different Areas

We evaluate the performance in two different office buildings to show the scalability of PYLON. Fig. 3.28 shows the topology and walking trace of each environment. Further experiments are performed on Google Pixel 3 because the performance of PYLON and the comparison schemes tends to be similar regardless of the type of smartphones. The experimental details are summarized in Table 1. We repeat the experiments 15 times for each trace on a round trip. Areas 1 and 2 represent office buildings 1 and 2, respectively. Sources indicated by small grey circles represent WiFi APs and BLE beacons. The number of rooms (#Rooms) means the number of areas that the user passes through. The number of doors (#Doors) indicates how many times the user has passed through the door. Fig. 3.29 depicts the performance of PYLON as well as three other comparison schemes in two different areas. PYLON achieves the best performance in the both areas. PYLON yields an average localization accuracy of 1.29 m and 1.11 m in office buildings 1 and 2, respectively. In the case of *LC* and PYLON, localization errors are independent of trace length. However, *Legacy* and *PF* have no landmark correction, so the longer the trace length, the larger the error. In conclusion, the experimental results show that PYLON achieves constant performance regardless of environmental changes.

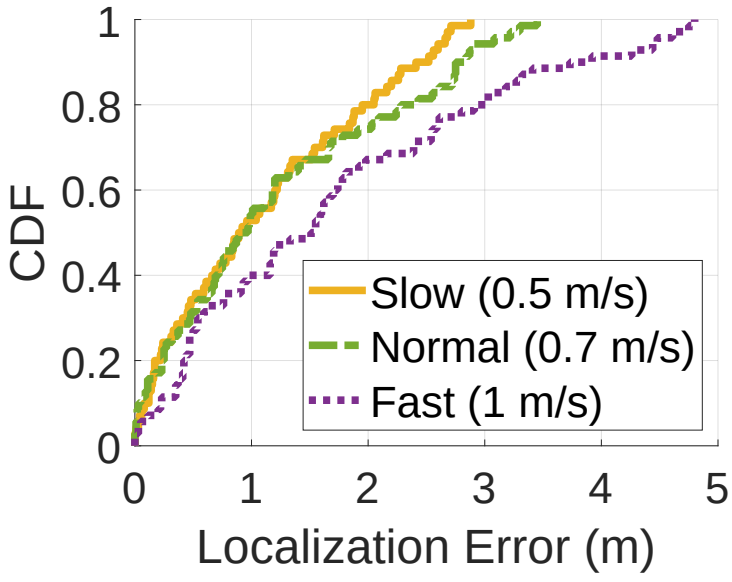
3.9 Summary

In this paper, we presented PYLON, a practical path estimation and localization system, that runs on a server and uses radio signal measurements performed on a smartphone. PYLON uses room doors as landmarks in indoor environments to realize path estimation and localization. In particular, it introduces a novel floor plan mapping algorithm that maps the relative positions of virtually generated rooms to the real-world floor plan. PYLON detects the exact time that the user passes through a door to compensate for accumulated IMU sensing errors of the smartphone. We implemented PYLON on

five Android smartphones, and evaluated its performance in an indoor office environment. We confirmed that PYLON achieves 97% floor plan mapping accuracy with a localization error of only 1.42 m on average.

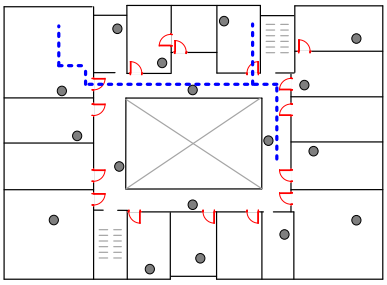


(a) Different walking distance

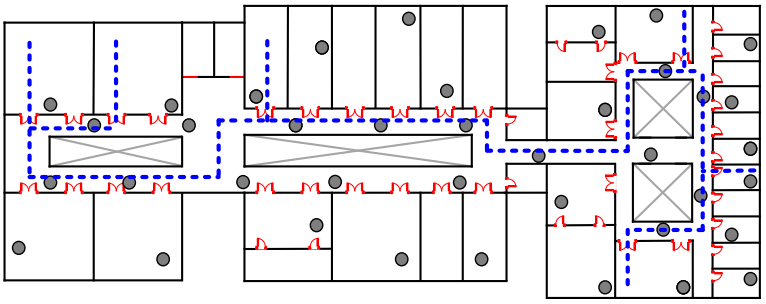


(b) Different walking speed

Figure 3.27: Impact of different walking distance and speed.



(a) Office building 1



(b) Office building 2

Figure 3.28: Two different experimental areas.

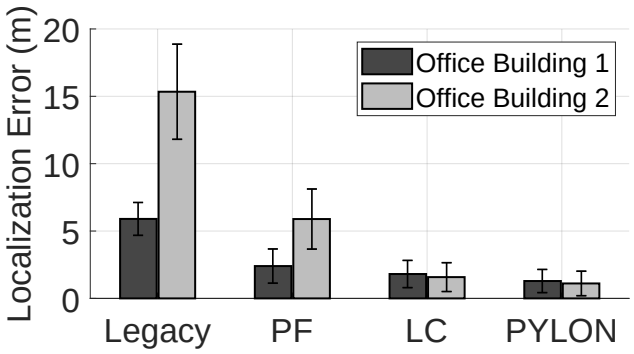


Figure 3.29: Localization performance on different areas.

Chapter 4

FINISH: Fully-automated Indoor Navigation using Smartphones with Zero Human Assistance

4.1 Introduction

Despite extensive research into indoor localization, the wide deployment of indoor navigation systems have yet to be realized. To bootstrap the indoor navigation services with zero human assistance, we ask the following question: *Can we enable users to easily bootstrap indoor navigation services without any assistance or intervention of users?* In this paper, we provide an answer through the systematic design and implementation of FINISH with the help of a real-world floor plan. Unlike other navigation system, FINISH does not require fingerprint data and assistance of the users. We have implemented FINISH on five different Android smartphones and evaluated it in the whole floors of an office building. Our experimental results show that FINISH achieves 100% initial location accuracy with in one step and provides timely navigation instructions.

In summary, the main contributions of this paper are threefold:

- We design a fully-automated navigation system, termed FINISH, which addresses the problems of existing previous indoor navigation systems. FINISH

generates the radio map of an indoor building based on the localization system to determine the initial location of the user.

- FINISH relaxes some requirements for current indoor navigation systems. It does not require any human assistance to provide navigation instructions. In addition, it is plug-and-play on diverse Android smartphones.

- We implement FINISH on five Android smartphones and evaluate it on five floors of an office building with the help of multiple users to prove applicability and scalability. FINISH determines the location of the user with extremely high accuracy with in one step.

4.2 Related Work

4.2.1 Localization-based Navigation System

Localization and Navigation have been extensively studied in the robotic area [29]. By fusing odometer outputs using IMU sensor readings, robots can compute travel distances, perform accurate localization, and navigate themselves to the destination based on the floor map. In several robotic systems, additional sensing techniques using laser [30], infrared [31], and camera [32] are also used for ranging and navigation purposes. However, the motion of human is more complicate, and the limited sensing capabilities of smartphones also challenge to both localization and navigation.

To address the aforementioned challenges, numerous localization techniques using smartphones have been proposed [33–40]. They can be broadly categorized as infrastructure-based (e.g, Wi-Fi, BLE, GPS, and cellular) or infrastructure-free (e.g., dead reckoning) localization system. Each of them has its own advantages and disadvantages. GPS can provide accurate positioning in outdoor open spaces but encounters fading signals in an indoor environments. Using radio signals, such as Wi-Fi and BLE, usually requires fingerprinting data to realize localization. In case of dead reckoning,

it suffers from cumulative errors and the usage of smartphones.

4.2.2 Peer-to-peer Navigation System

The navigation systems using leader-follower model have been proposed [29, 41–43]. An electronic escort system was proposed by using crowd encounters information and dead-reckoning techniques [41], but it requires pre-deployed audio beacons which limits its applicability. A vision-guided navigation system, termed Travi-Navi [42], enables a user to easily bootstrap and deploy indoor navigation system without help of indoor localization system, but guiders need to hold the smartphone vertically and steadily during walking to achieve a better image quality. FollowMe [29] uses compute-intensive particle filtering as the navigation engine, and minimizes the constraints imposed on users by providing wider usage (i.e., in multi-level buildings, semi-outdoors). FollowUs [43] is incrementally-deployable navigation by automatically generating the trace with the data of multiple users. However, above two navigation systems require assistance of user. The leader needs to input the initial location and destination to provide navigation service, which is vulnerable to the false input of the leaders of malicious users.

4.3 System Overview

This section provides an overview of `FINISH`, first presenting the system architecture and then a navigation example to illustrate how `FINISH` operates.

4.3.1 System Architecture

Basically, `FINISH` operates on the RSSI-based localization system [40]. This localization system can collect RSSI signatures of each room and corridor from multiple radio sources in indoor environment without human intervention. We borrow this system to utilize RSSI signatures and to provide indoor navigation service for smartphones with-

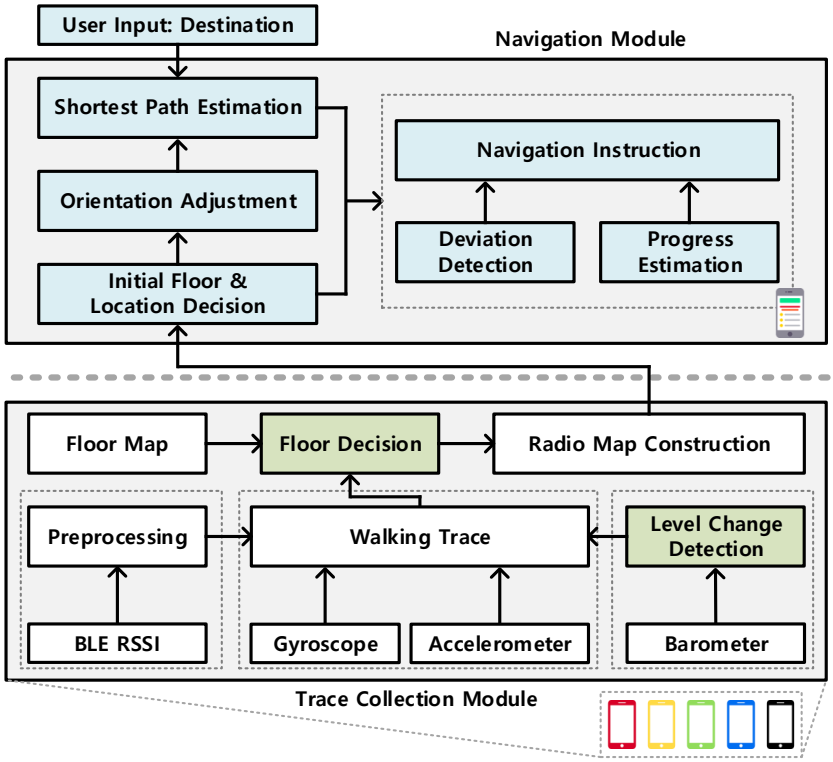
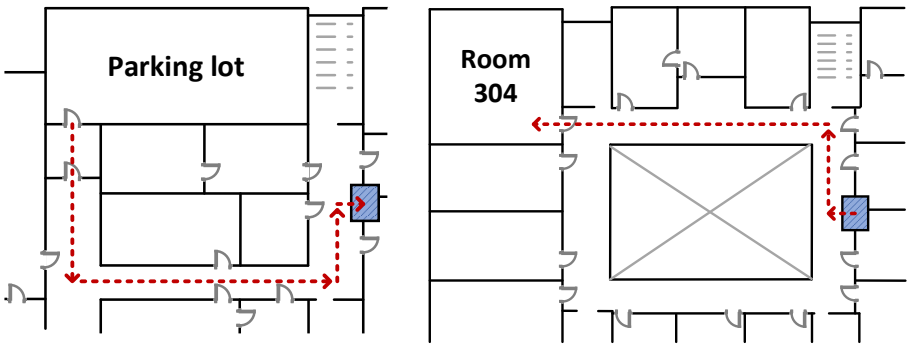


Figure 4.1: Architecture of FINISH.

out the assistance of users. However, the localization system does not consider multi-floor operation that is essential to indoor navigation service. Therefore, we leverage pressure sensor (i.e., barometer) to detect the level-change for multi-floor operation. The barometer is hardware-based sensor that returns ambient air pressure value in hPa.

FINISH consists of a trace collection module and a navigation module, and both modules exploit RSSI signature and multiple sensors in a smartphones, such as accelerometer, gyroscope, and barometer.

Fig 4.1 shows the overall architecture of FINISH.

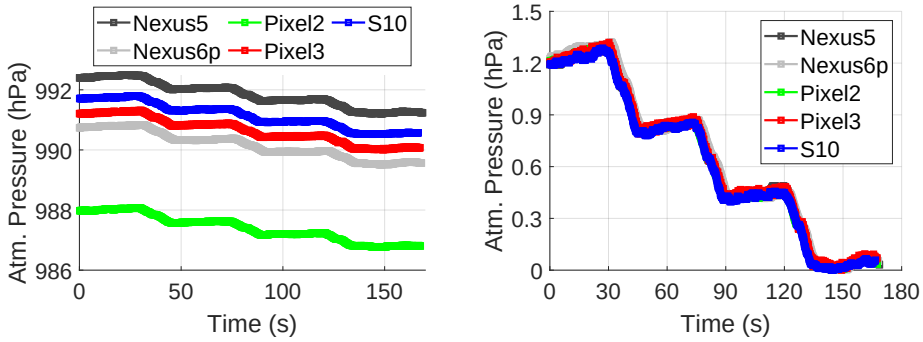


(a) The first basement floor (b) The 3rd floor

Figure 4.2: An indoor navigation example.

4.3.2 An Example for Navigation

We take an example of indoor navigation for easy understanding as shown in Fig 4.2. For example, if a meeting will be held in Room 304 on the third floor of our office building, the visitor needs indoor navigation from the parking lot on the first basement floor. When the visitor enters into the building from the parking lot, `FINISH` quickly determines the coarse location of the visitor. If the visitor enters the destination as "304", the navigation module calculates the shortest path from the visitor's location. The navigation module guides the visitor to the area that an elevator is located. `FINISH` constantly tracks the location of the visitor and detects whether the visitor takes the elevator or not. If the visitor takes the elevator and get off on the third floor, navigation module gives instructions to the Room 304. If `FINISH` considers that the visitor arrives at the destination, it finishes the navigation.



(a) Barometer readings

(b) Barometer differences

Figure 4.3: Example of barometer readings for level-change detection.

4.4 Level Change Detection and Floor Decision

4.4.1 Level Change Detection

Note that the localization system that we exploit does not consider multi-floor operation. Therefore, `FINISH` needs to detect the level-change of user's movements from the initial data collection stage, and we adopt a barometer in smartphones for level-change detection. We conduct preliminary experiment to understand the nature of the barometer in different smartphones.

We take a walk using five different smartphones at the same time from the first basement floor to the third floor using stairs. As shown in Fig. 4.3a, barometer readings decrease three time because an atmospheric pressure decreases as an altitude increases. The barometer readings show the similar trend regardless of device heterogeneity, but the absolute values have offset between devices up to 5 hPa. Therefore, using the absolute barometer value of each device is not practical to detect the level change considering the device heterogeneity. However, we discover that the gap of barometer reading maintains for each floor and independents to the device type as shown in Fig. 4.3b. We calculate the atmospheric pressure p gap between floor using the standard pressure-height formula [112]:

$$h = 44,330 \cdot (1 - (\frac{p}{p_0})^{1/5.255}) \tag{4.1}$$

where h is the altitude in meter, while p and p_0 are the measured pressure and sea level pressure respectively in hPa. The sea level reference does not matter, since we only use relative height change. Following the Equation 4.1, we can obtain 0.4 hPa gap between floors if the height difference between floor is considered 3.5 m.

In FINISH, we devise level-change detection algorithm based on the aforementioned experiments and formula. FINISH first smooths each pressure p_n by passing low-pass filter. The algorithm then tracks the maximal difference between p_n and samples collected with in the last T_0 seconds. If the $|p_n - p_m|$ is greater than the threshold p_{th}, we consider that level-change is happened. The algorithm records a value f_l that indicates whether the altitude increase/decrease $(f_l = -1/1)$ or not $(f_l = 0)$. In particularly, if the step frequency doubles up during the level-change period, we record an elevator up/down, otherwise we record an stair up/down.

4.5 Real-time navigation

4.5.1 Initial Floor and Location Decision

After collecting traces of users on multiple floor of a building, we have to determine the initial location of the user to provide real-time indoor navigation service. We can obtain the RSSI stacking difference data of each room when generating the virtual rooms. Therefore, FINISH can determine the initial location of the user by comparing the RSSI stacking difference values between rooms and smartphone. We denote the RSSI stacking difference of room as

$$W_i = [w_{i1}, w_{i2}, \cdots, w_{im}], \tag{4.2}$$

where m represents the number of beacons. Similarly, the RSSI stacking difference instantly collected on the smartphone can be represented as

$$W = [w_1, w_2, \cdots, w_l], \tag{4.3}$$

where l means the number of beacons that the smartphone can receive the packet. Since both RSSI stacking differences of rooms and smartphone are non-zero vectors, we use *cosine similarity* to determine the initial location of the user. Cosine similarity is a measure for similarity between two non-zero vectors of an inner product space, which measures the cosine of the angle between them [110]. It is not a magnitude but the judgment of orientation. The cosine similarity has a value between -1 and 1. If two vectors have the same orientation, the cosine similarity is 1. Two vectors in diametrically opposite orientation have the cosine similarity of -1, independent of their magnitude. We can represent the cosine similarity *sim* of two vectors W_i and W as

$$sim(W_i, W) = \frac{W_i \cdot W}{||W_i||||W||}. \tag{4.4}$$

`FINISH` determines the initial location of the user to the room that shows the highest *sim*.

4.5.2 Orientation Adjustment

After finding the user's location, the estimating the walking direction is important to provide navigation instruction. `FINISH` uses *sim* to estimate the orientation of the user. When the initial location is determined, we compare the *sim* of each room. Therefore, *sim*s of neighbor rooms of the initial location also can be obtained in the process of initial location decision. In addition, the gradients of *sim* value of neighbor rooms are collected in the walking progress of the user. We adjust the orientation of the user as the direction from the initial location to the room that shows the highest gradient value.

4.5.3 Shortest Path Estimation

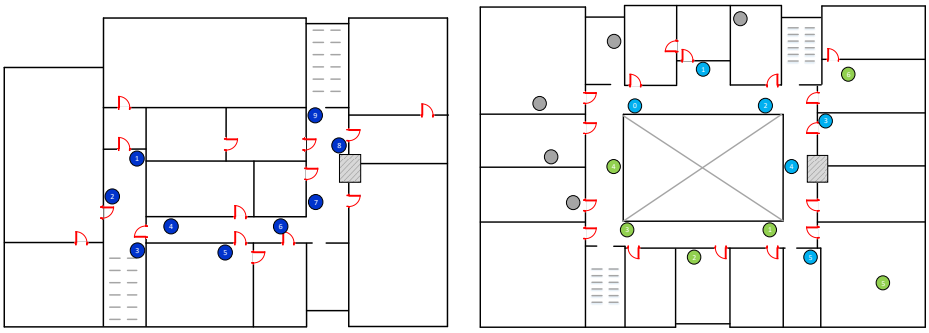
Now, we have the knowledge of the initial location and the orientation of the user, and estimate the shortest path to the destination. The destination information is an input from the user. FINISH represents the indoor map as a graph from the radio map generation process. Therefore, the node and cost of the graph can be considered room and distance between rooms. In graph theory, finding the shortest path is well-known problem, and Dijkstra algorithm is one of the most popular solution. We adopt Dijkstra algorithm to estimate the shortest path from the initial location to the destination.

4.6 Performance Evaluation

We conduct experiments for total test area $7,270\ m^2$ in an office building, and use only BLE beacons. Fig. 4.4 shows the example of deployment of BLE beacons, which are represented by small circles. Each BLE beacon periodically transmits BLE advertising packets every 100 ms that equals the WiFi AP beacon interval. We develop FINISH as an application on the Android OS of five smartphones to cover device heterogeneity: Nexus 5, Nexus 6p, Pixel 2, Pixel 3, and Galaxy S10. Since Nexus 5 was released in 2013 and Galaxy S10 in 2019, we experimented with the smartphones that have been released in the last 7 years. The application is plug-and-play and collects RSSI data of BLE, gyroscope, accelerometer, and barometer.

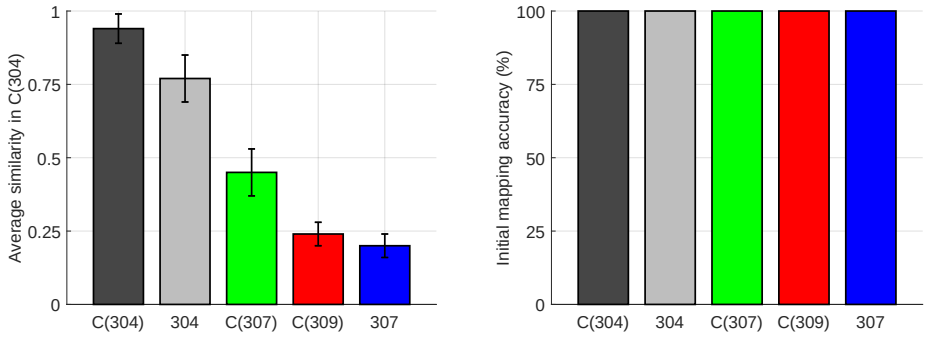
4.6.1 Initial Location Accuracy

We first evaluate the performance of initial location accuracy on five different area. We conduct experiment 20 times with Google Pixel 3. Fig. 4.5a shows the average similarity of areas when the user is located in C(304). The highest similarity shows in C(304) and the similarities of other neighbor ares are lower that C(304). Therefore, the similarity can successfully differentiate the location of the user. Fig. 4.5 shows the mapping accuracy of the initial location of the user. The x-axis represents the initial



(a) The first basement floor (b) The 3rd floor

Figure 4.4: The node placement of building.



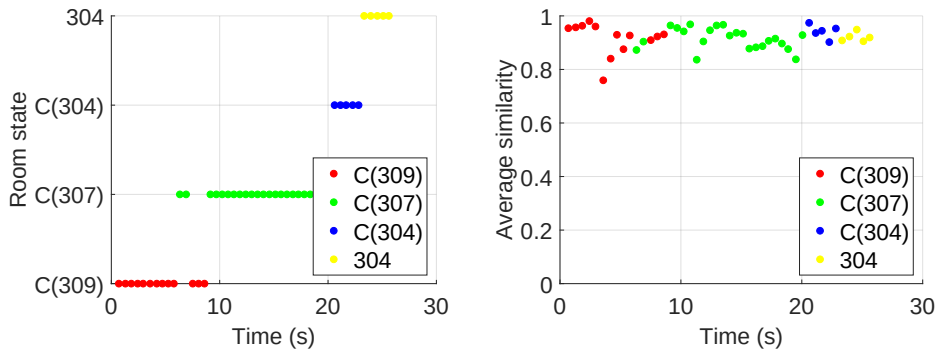
(a) Average similarity from C(304) (b) The initial location accuracy

Figure 4.5: Estimation accuracy of initial location.

location of the user, and the y-axis means the ratio of correctly estimate the initial location. In five locations, FINISH successfully estimate the initial location of the user with 100% accuracy.

4.6.2 Real-Time Navigation Accuracy

We evaluate the performance of real-time navigation accuracy with room state change and average similarity. While the user walks from C(309) to 304, FINISH tracks the walking progress by estimating the real-time location of the user. As shown in Fig. 4.6,



(a) Room state change during navigation

(b) Average similarity during navigation

Figure 4.6: Room state change and average similarity.

FINISH successfully estimates the location of the user every steps until the user arrives at the destination 304. During the room state transition, the average similarity between the room and smartphone maintains over 0.8, which means the FINISH can follow the room state change with high similarity.

4.7 Summary

In this paper, we presented FINISH, an fully-automated indoor navigation system with zero human assistance, that runs plug-and-play on smartphones. FINISH uses barometer in level change detection to realize the operation in multiple floor, and generates the radio map based on the localization system. In particular, it introduce a novel initial floor and location detection algorithm to completely exclude user intervention. FINISH determines the exact initial location of the user and provides timely instruction to the desired destination. We implemented FINISH on five Android smartphones, and evaluated its performance in an office environment. We confirmed that FINISH achieves 100% accuracy of initial location with in one step of the user.

Chapter 5

Conclusion

5.1 Research Contributions

In the dissertation, we have addressed the systems that improve the user experience for smartphones using wireless communication technologies, especially Wi-Fi and BLE.

In Chapter 2, we have proposed BLEND, BLE beacon-aided fast Wi-Fi handoff for smartphones. We conduct detailed analysis of the sticky client problem on commercial smartphones with experiment and close examination of Android source code. We propose BLEND, which exploits BLE modules to provide smartphones with prior knowledge of the presence and information of APs operating at both 2.4 and 5 GHz Wi-Fi channels. BLEND operating with only application requires no hardware and Android source code modification of smartphone. To our best knowledge, BLEND is the first BLE-aided handoff scheme. We also propose an advanced version of BLEND that can be applied to smartphone enabling hidden Android API, which optimizes Wi-Fi scanning through modification of Android source code. We prototype BLEND with commercial smartphones and evaluate the performance in real environments. Our measurement results demonstrate that BLEND significantly improves throughput and video bitrate by up to 61% and 111%, compared to a commercial Android application, respectively, with negligible energy overhead.

In Chapter 3, we have presented PYLON, smartphone based indoor path estimation and localization without human intervention. We design a path estimation and localization system, termed PYLON, which is plug-and-play on Android smartphones. PYLON includes a novel landmark correction scheme that leverages real doors of indoor environments consisting of floor plan mapping, door passing time detection and correction. It operates without any user intervention. PYLON relaxes some requirements for localization systems. It does not require any modifications to hardware or software of smartphones, and the initial location of Wi-Fi APs, BLE beacons, and users. In addition, on-site investigations for fingerprint or trace data collection which are labor-intensive and time-consuming is not necessary. PYLON can be directly applied to unknown indoor environments. We implement PYLON on five Android smartphones and evaluate it on two office buildings with the help of three participants to prove applicability and scalability. PYLON achieves very high floor plan mapping accuracy with a low localization error.

In Chapter 4, we present FINISH, fully-automated indoor navigation using smartphones with zero human assistance. FINISH generates the radio map of an indoor building based on the localization system to determine the initial location of the user. FINISH relaxes some requirements for current indoor navigation systems. It does not require any human assistance to provide navigation instructions. We implement FINISH on five Android smartphones and evaluate it on five floors of an office building with the help of multiple users to prove applicability and scalability.

5.2 Future Work

As further improvement on the results of this dissertation, there are two research items regarding indoor navigation system. First, we will extend the experiment area to whole floors of an office building by constructing the testbed. Second, we will automatically generate the physical graph by using image processing techniques.

Bibliography

[1] J. Shi, L. Meng, A. Striegel, C. Qiao, D. Koutsonikolas, and G. Challen, "A Walk on the Client Side: Monitoring Enterprise WiFi Networks Using Smartphone Channel Scans," in *Proc. IEEE INFOCOM*, 2016.

[2] I. Ramani and S. Savage, "SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks," in *Proc. IEEE INFOCOM*, 2005.

[3] Y. Chen, M.-C. Chuang, and C. Chen, "DeuceScan: Deuce-based Fast Handoff Scheme in IEEE 802.11 Wireless Networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 2, 2008.

[4] J. Teng, C. Xu, W. Jia, and D. Xuan, "D-Scan: Enabling Fast and Smooth Handoffs in AP-dense 802.11 Wireless Networks," in *Proc. IEEE INFOCOM*, 2009.

[5] H. Wu, K. Tan, Y. Zhang, and Q. Zhang, "Proactive Scan: Fast Handoff with Smart Triggers for 802.11 Wireless LAN," in *Proc. IEEE INFOCOM*, 2007.

[6] P. Lv, X. Wang, X. Xue, and M. Xu, "SWIMMING: Seamless and Efficient WiFi-based Internet Access from Moving Vehicles," *IEEE Transactions on Mobile Computing*, vol. 14, no. 5, 2015.

[7] Z. Song, L. Shangguan, and K. Jamieson, "WiFi Goes to Town: Rapid Picocell Switching for Wireless Transit Networks," in *Proc. ACM SIGCOMM*, 2017.

[8] IEEE 802.11, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Radio Resource Measurement of Wireless LANs.*, IEEE Std., 2008.

[9] IEEE 802.11, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Fast Basic Service Set (BSS) Transition.*, IEEE Std., 2008.

[10] M. I. Sanchez and A. Boukerche, "On IEEE 802.11 k/r/v Amendments: Do They Have a Real Impact?" *IEEE Wireless Communications*, vol. 23, no. 1, 2016.

[11] R. Chandra and P. Bahl, "MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card," in *Proc. IEEE INFOCOM*, 2004.

[12] A. J. Nicholson, S. Wolchok, and B. D. Noble, "Juggler: Virtual Networks for Fun and Profit," *IEEE Transactions on Mobile Computing*, vol. 9, no. 1, 2010.

[13] S. Kandula, K. C. Lin, T. Badirkhanli, and D. Katabi, "FatVAP: Aggregating AP Backhaul Capacity to Maximize Throughput," in *Proc. USENIX NSDI*, 2008.

[14] A. Croitoru, D. Niculescu, and C. Raiciu, "Towards WiFi Mobility without Fast Handover," in *Proc. USENIX NSDI*, 2015.

[15] B. Vladimir, M. Arunesh, and B. Suman, "Eliminating Handoff Latencies in 802.11 WLANs Using Multiple Radios: Applications, Experience, and Evaluation," in *Proc. ACM IMC*, 2005.

[16] S. Jin, M. Choi, and S. Choi, "Multiple WNIC-based Handoff in IEEE 802.11 WLANs," *IEEE Communications Letters*, vol. 13, no. 10, 2009.

[17] S. Jin and S. Choi, "A Seamless Handoff with Multiple Radios in IEEE 802.11 WLANs," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 3, 2014.

[18] W. Kang and Y. Han, "SmartPDR: Smartphone-based Pedestrian Dead Reckoning for Indoor Localization," *IEEE Sensors Journal*, vol. 15, no. 5, pp. 2906–2916, 2014.

[19] P. Zhou, M. Li, and G. Shen, "Use It Free: Instantly Knowing Your Phone Attitude," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking (MobiCom)*. Maui, HI, USA: ACM, 2014, pp. 605–616.

[20] A. T. Mariakakis, S. Sen, J. Lee, and K.-H. Kim, "SAIL: Single Access Point-based Indoor Localization," in *Proceedings of the 12th International Conference on Mobile Systems, Applications, and Services (MobiSys)*. Bretton Woods, NH, USA: ACM, 2014, pp. 315–328.

[21] B. Zhou, Q. Li, Q. Mao, W. Tu, X. Zhang, and L. Chen, "ALIMC: Activity Landmark-based Indoor Mapping via Crowdsourcing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2774–2785, 2015.

[22] S. He, S.-H. G. Chan, L. Yu, and N. Liu, "Calibration-free Fusion of Step Counter and Wireless Fingerprints for Indoor Localization," in *Proceedings of the 17th International Conference on Ubiquitous Computing (UbiComp)*. Osaka, Japan: ACM, 2015, pp. 897–908.

[23] S. Beauregard and H. Haas, "Pedestrian Dead Reckoning: A Basis for Personal Positioning," in *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication (WPNC)*. Hannover, Germany: Shaker Verlag, 2006, pp. 27–35.

[24] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, "Walkie-Markie: Indoor Pathway Mapping made Easy," in *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. Washington, D.C, USA: USENIX, 2013, pp. 85–98.

[25] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No Need to War-drive: Unsupervised Indoor Localization," in *Proceedings of the 10th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*. Low Wood Bay, Lake District, UK: ACM, 2012, pp. 197–210.

[26] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li, "Jigsaw: Indoor Floor Plan Reconstruction via Mobile Crowdsensing," in *Proc. ACM MobiCom*, 2014.

[27] C. Wu, Z. Yang, and C. Xiao, "Automatic Radio Map Adaptation for Indoor Localization using Smartphones," *IEEE Transactions on Mobile Computing*, vol. 17, no. 3, pp. 517–528, 2017.

[28] J. Xu, H. Chen, K. Qian, E. Dong, M. Sun, C. Wu, L. Zhang, and Z. Yang, "iVR: Integrated Vision and Radio Localization with Zero Human Effort," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 3, no. 3, pp. 1–22, 2019.

[29] Y. Shu, K. G. Shin, T. He, and J. Chen, "Last-mile Navigation using Smartphones," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom)*. Paris, France: ACM, 2015, pp. 512–524.

[30] S. H. Cho and S. Hong, "Map based indoor robot navigation and localization using laser range finder," in *Proc. IEEE ICARCV*, 2010.

[31] A. M. Flynn, "Combining Sonar and Infrared Sensors for Mobile Robot Navigation," *Int. J. Rob. Res.*, vol. 7, no. 6, pp. 5–14, 1988.

[32] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual Navigation for Mobile Robots: A Survey," *Journal of Intelligent and Robotic Systems*, vol. 53, no. 3, p. 263, 2008.

[33] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor Localization without the Pain," in *Proceedings of the 16th Annual International Conference on Mobile Computing and Networking (MobiCom)*. Chicago, Illinois, USA: ACM, 2010, pp. 173–184.

[34] I. Constandache, R. R. Choudhury, and I. Rhee, "Towards mobile phone localization without war-driving," in *Proc. IEEE INFOCOM*, 2010.

[35] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao, "A reliable and accurate indoor localization method using phone inertial sensors," in *Proceedings of the 14th International Conference on Ubiquitous Computing (UbiComp)*. Pittsburgh, PA, USA: ACM, 2012, pp. 421–430.

[36] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort Crowdsourcing for Indoor Localization," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom)*. Istanbul, Turkey: ACM, 2012, pp. 293–304.

[37] Y. Chen, D. Lymberopoulos, J. Liu, and B. Priyantha, "FM-based Indoor Localization," in *Proc. ACM MobiSys*, 2012.

[38] J. Xiong and K. Jamieson, "Arraytrack: A Fine-grained Indoor Location System," in *Proc. USENIX NSDI*, 2013.

[39] Y. Shu, P. Coué, Y. Huang, J. Zhang, P. Cheng, and J. Chen, "G-Loc: Indoor localization leveraging gradient-based fingerprint map," in *Proc. IEEE INFOCOM Workshop*, 2014.

[40] J. Choi, G. Lee, S. Choi, and S. Bahk, "Smartphone based Indoor Path Estimation and Localization without Human Intervention," in *under review*, 2020.

[41] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury, "Did you see Bob? Human Localization using Mobile Phones," in *Proc. ACM MobiCom*, 2010.

[42] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao, "Travi-Navi: Self-deployable Indoor Navigation System," in *Proc. ACM MobiCom*, 2014.

[43] Y. Shu, Z. Li, B. Karlsson, Y. Lin, T. Moscibroda, and K. Shin, "Incrementally-deployable Indoor Navigation with Automatic Trace Generation," in *Proc. IEEE INFOCOM*, 2019.

[44] Cisco Visual Networking Index, "Global Mobile Data Traffic Forecast Update, 2016–2021," *White paper*, 2017.

[45] X. Chen and D. Qiao, "HaND: Fast Handoff with Null Dwell Time for IEEE 802.11 Networks," in *Proc. IEEE INFOCOM*, 2010.

[46] J. Choi, "WidthSense: WiFi Discovery via Distance-based Correlation Analysis," *IEEE Communications Letters*, vol. 21, no. 2, 2017.

[47] Y. Xiong, R. Zhou, M. Li, G. Xing, L. Sun, and J. Ma, "ZiFi: Exploiting Cross-Technology Interference Signatures for Wireless LAN Discovery," *IEEE Transactions on Mobile Computing*, vol. 13, no. 11, 2014.

[48] Y. Gao, J. Niu, R. Zhou, and G. Xing, "ZiFind: Exploiting Cross-Technology Interference Signatures for Energy-Efficient Indoor Localization," in *Proc. IEEE INFOCOM*, 2013.

[49] G. Ananthanrayanan and I. Stocia, "Blue-Fi: Enhancing WiFi Performance Using Bluetooth Signals," in *Proc. ACM MobiSys*, 2009.

[50] Samsung Smart Home website http://www.samsung.com/us/explore/connect/.

[51] Google WiFi AP website https://madeby.google.com/wifi/.

[52] Ubertooth One website http://ubertooth.sourceforge.net/hardware/one/.

[53] A. Mishra, M. Shin, and W. Arbaugh, "An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, 2003.

[54] IEEE 802.11, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements.*, IEEE Std., 2005.

[55] *Specification of the Bluetooth System, Covered Core Package Version: 4.0*, The Bluetooth Special Interest Group (SIG), 2010.

[56] Android Open Source Project https://source.android.com/.

[57] Riverbed AirPcap website. https://support.riverbed.com/content/support/software/steelcentral-npm/airpcap.html.

[58] Google Beacon Platform website. https://developers.google.com/beacons/.

[59] BCM4339 Datasheet website. http://www.cypress.com/documentation/datasheets/cyw4339-single-chip-5g-wifi-ieee-80211ac-macbasebandradio-integrated.

[60] V. Mhatre and K. Papagiannaki, "Using Smart Triggers for Improved User Performance in 802.11 Wireless Networks," in *Proc. ACM MobiSys*, 2006.

[61] Linux kernel backports website. http://drvbp1.linux-foundation.org/~mcgrof/rel-html/backports/.

[62] Google ExoPlayer website. https://google.github.io/ExoPlayer/.

[63] FFmpeg website. https://www.ffmpeg.org/.

[64] Top 5 Android WiFi Manager website. https://tunesgo.wondershare.com/android/ android-wifi-manager.html.

[65] Monsoon Solution Incorporate website.
https://www.msoon.com/LabEquipment/.

[66] S. He and S.-H. G. Chan, "WiFi Fingerprint-based Indoor Positioning: Recent Advances and Comparisons," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 466–490, 2015.

[67] P. Bahl, V. N. Padmanabhan, V. Bahl, and V. Padmanabhan, "RADAR: An in-building RF-based User Location and Tracking System," in *Proceedings of the 20th Annual IEEE International Conference on Computer Communications (INFOCOM)*. Anchorage, AK, USA: IEEE, 2000.

[68] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Practical Robust Localization over Large-scale 802.11 Wireless Networks," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom)*. Philadelphia, PA, USA: ACM, 2004, pp. 70–84.

[69] J.-g. Park, B. Charrow, D. Curtis, J. Battat, E. Minkov, J. Hicks, S. Teller, and J. Ledlie, "Growing an Organic Indoor Location System," in *Proceedings of the 8rd International Conference on Mobile Systems, Applications, and Services (MobiSys)*. San Francisco, CA, USA: ACM, 2010, pp. 271–284.

[70] A. Williams, D. Ganesan, and A. Hanson, "Aging in Place: Fall Detection and Localization in a Distributed Smart Camera Network," in *Proceedings of the 15th ACM International Conference on Multimedia*. Augsburg, Germany: ACM, 2007, pp. 892–901.

[71] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-support System," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom)*. Bostan, MA, USA: ACM, 2000, pp. 32–43.

[72] Z. Sun, A. Purohit, K. Chen, S. Pan, T. Pering, and P. Zhang, "PANDAA: Physical Arrangement Detection of Networked Devices through Ambient-sound Awareness," in *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp)*. Beijing, China: ACM, 2011, pp. 425–434.

[73] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye, "Push the Limit of WiFi based Localization for Smartphones," in *Proc. ACM MobiCom*, 2012.

[74] W. Huang, Y. Xiong, X.-Y. Li, H. Lin, X. Mao, P. Yang, and Y. Liu, "Shake and Walk: Acoustic Direction Finding and Fine-grained Indoor Localization using Smartphones," in *Proceedings of the 33rd Annual IEEE International Conference on Computer Communications (INFOCOM)*. Toronto, Canada: IEEE, 2014, pp. 370–378.

[75] D. Hauschildt and N. Kirchhof, "Advances in Thermal Infrared Localization: Challenges and Solutions," in *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. Zurich, Switzerland: IEEE, 2010, pp. 1–8.

[76] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, "Spotfi: Decimeter Level Localization using Wifi," in *Proc. ACM SIGCOMM*, 2015.

[77] J. Wang, H. Jiang, J. Xiong, K. Jamieson, X. Chen, D. Fang, and B. Xie, "LiFS: Low Human-effort, Device-free Localization with Fine-grained Subcarrier Information," in *Proc. ACM MobiCom*, 2016.

[78] F. Lemic, J. Martin, C. Yarp, D. Chan, V. Handziski, R. Brodersen, G. Fettweis, A. Wolisz, and J. Wawrzynek, "Localization as a Feature of mmWave Communication," in *Proceedings of the 12th International Wireless Communications and Mobile Computing Conference (IWCMC)*. Cyprus, Paphos: IEEE, 2016, pp. 1033–1038.

[79] IEEE 802.11ad, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 3: Enhancements for Very High Throughput in the* 60 *GHz Band.*, IEEE Std., 2012.

[80] C. Zhang and X. Zhang, "LiTell: Robust Indoor Localization using Unmodified Light Fixtures," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking (MobiCom)*. New York City, NY, USA: ACM, 2016, pp. 230–242.

[81] Y.-S. Kuo, P. Pannuto, K.-J. Hsiao, and P. Dutta, "Luxapose: Indoor Positioning with Mobile Phones and Visible Light," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking (MobiCom)*. Maui, HI, USA: ACM, 2014, pp. 447–458.

[82] L. Li, P. Hu, C. Peng, G. Shen, and F. Zhao, "Epsilon: A visible light based positioning system," in *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. San Diego, CA, USA: USENIX, 2014, pp. 331–343.

[83] Z. Zhao, J. Wang, X. Zhao, C. Peng, Q. Guo, and B. Wu, "NaviLight: Indoor Localization and Navigation under Arbitrary Lights," in *Proceedings of the 36th Annual IEEE International Conference on Computer Communications (INFOCOM)*. Atlanta, GA, USA: IEEE, 2017, pp. 1–9.

[84] Z. Yang, C. Wu, and Y. Liu, "Locating in Fingerprint Space: Wireless Indoor Localization with Little Human Intervention," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (MobiCom)*. Istanbul, Turkey: ACM, 2012, pp. 269–280.

[85] Y.-C. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm, "Accuracy Characterization for Metropolitan-scale Wi-Fi Localization," in *Proceedings of the 3rd*

International Conference on Mobile Systems, Applications, and Services (MobiSys). Seattle, WA, USA: ACM, 2005, pp. 233–245.

[86] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "LANDMARC: Indoor Location Sensing using Active RFID," in *Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom)*. Fort Worth, TX, USA: IEEE, 2003, pp. 407–415.

[87] C. Wu, Z. Yang, and Y. Liu, "Smartphones Based Crowdsourcing for Indoor Localization," *IEEE Transactions on Mobile Computing*, vol. 14, no. 2, pp. 444–457, 2014.

[88] M. Youssef and A. Agrawala, "The Horus WLAN Location Determination System," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services (MobiSys)*. Seattle, WA, USA: ACM, 2005, pp. 205–218.

[89] M. Azizyan, I. Constandache, and R. Roy Choudhury, "SurroundSense: Mobile Phone Localization via Ambience Fingerprinting," in *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom)*. Beijing, China: ACM, 2009, pp. 261–272.

[90] H. Lim, L.-C. Kung, J. C. Hou, and H. Luo, "Zero-configuration Indoor Localization over IEEE 802.11 Wireless Infrastructure," *Wireless Networks*, vol. 16, no. 2, pp. 405–420, 2010.

[91] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 1996, vol. 2.

[92] H. Lim, L.-C. Kung, J. C. Hou, and H. Luo, "Zero-configuration, Robust Indoor Localization: Theory and Experimentation," in *Proceedings of the 25th Annual IEEE International Conference on Computer Communications (INFOCOM)*. Barcelona, Catalunya, Spain: IEEE, 2006.

[93] M. Baianifar, S. M. Razavizadeh, H. Akhlaghpasand, and I. Lee, "Energy Efficiency Maximization in mmWave Wireless Networks with 3D Beamforming," *Journal of Communications and Networks*, vol. 21, no. 2, pp. 125–135, 2019.

[94] D. Turner, S. Savage, and A. C. Snoeren, "On the Empirical Performance of Self-calibrating Wifi Location Systems," in *2011 IEEE 36th Conference on Local Computer Networks (LCN)*. Bonn, Germany: IEEE, 2011, pp. 76–84.

[95] R. C. Smith and P. Cheeseman, "On the Representation and Estimation of Spatial Uncertainty," *The international journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[96] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous Map Building and Localization for an Autonomous Mobile Robot," in *Proceedings IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS)*. Osaka, Japan: IEEE, 1991, pp. 1442–1447.

[97] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," in *Proceedings Association for the Advancement of Artificial Intelligence (AAAI)*. Edmonton, Alberta, Canada: AAAI, 2002, pp. 593–598.

[98] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT press, 2005.

[99] D. Amarasinghe, G. K. Mann, and R. G. Gosine, "Landmark Detection and Localization for Mobile Robot Applications: A Multisensor Approach," *Robotica*, vol. 28, no. 5, pp. 663–673, 2010.

[100] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," in *Proceedings Association for the Advancement of Artificial Intelligence (AAAI)*. Orlando, FL, USA: AAAI, 1999, pp. 343–349.

[101] P. Robertson, M. Angermann, and B. Krach, "Simultaneous Localization and Mapping for Pedestrians using only Foot-mounted Inertial Sensors," in *Proceedings of the 11th International Conference on Ubiquitous Computing (UbiComp)*. Orlando, FL, USA: ACM, 2009, pp. 93–96.

[102] B. Ferris, D. Fox, and N. D. Lawrence, "Wifi-slam using Gaussian Process Latent Variable Models," in *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI)*. Hyderabad, India: IJCAI, 2007, pp. 2480–2485.

[103] T. Pulkkinen, T. Roos, and P. Myllymäki, "Semi-supervised Learning for Wlan Positioning," in *Proceedings of the 28h International Conference on Artificial Neural Networks (ICANN)*. Espoo, Finland: Springer, 2011, pp. 355–362.

[104] J. Choi, G. Lee, Y. Shin, J. Koo, M. Jang, and S. Choi, "BLEND: BLE Beacon-aided Fast WiFi Handoff for Smartphones," in *Proceedings of the 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. Hong Kong: IEEE, 2018, pp. 1–9.

[105] C. Wu, Z. Yang, Y. Liu, and Y. Xi, "WILL: Wireless Indoor Localization without Site Survey," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 4, pp. 839–848, 2013.

[106] N. Roy, H. Wang, and R. Roy Choudhury, "I am a Smartphone and I Can Tell My User's Walking Direction," in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*. Bretton Woods, NH, USA: IEEE, 2014, pp. 329–342.

[107] W. Zijlstra, "Assessment of Spatio-temporal Parameters during Unconstrained Walking," *European journal of applied physiology*, vol. 92, no. 1-2, pp. 39–44, 2004.

[108] T. M. Kodinariya and P. R. Makwana, "Review on determining number of cluster in k-means clustering," *International Journal*, vol. 1, no. 6, pp. 90–95, 2013.

[109] L. C. Freeman, "A Set of Measures of Centrality based on Betweenness," *Sociometry*, vol. 40, pp. 35–41, 1977.

[110] A. Singhal, "Modern Information Retrieval: A Brief Overview," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 35–43, 2001.

[111] Estimote, "Estimote website," 2012. [Online]. Available: https://estimote.com

[112] K. Sankaran, M. Zhu, X. F. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh, "Using Mobile Phone Barometer for Low-power Transportation Context Detection," in *Proc. ACM SenSys*, 2014.

초 록

최근 스마트폰에 Wi-Fi나 BLE와 같은 무선통신기술 뿐만 아니라 다양한 센서가 탑재되고 있다. 또한, 많은 경우에 사용자들이 이동 중에 스마트폰을 활용하기 때문에, 이동성이 있는 사용자에게 무선통신기술과 다양한 센서를 활용한다면 더 나은 사용자 경험을 제공할 수 있다. 예를 들면, 사용자가 Wi-Fi를 활용하면서 이동할 때, 끊김없는 Wi-Fi 서비스를 제공하여 사용자 경험을 향상시킬 수 있다. 또한, 실내 환경에서 사용자의 이동성를 예측하여 실내 측위 기술이나 네비게이션과 같은 특별한 서비스를 제공할 수 있고, 위치 기반의 광고 및 결제 시스템과 같은 부가적인 서비스 또한 제공 가능하다. 따라서, 무선통신기술 및 스마트폰의 센서를 활용하여 사용자 경험의 향상은 앞으로 매우 중요한 연구분야라고 생각된다.

본 논문에서는 Wi-Fi와 BLE, 스마트폰의 센서를 활용하여 사용자의 경험 또는 편의성을 향상시킬 수 있는 세 가지 시스템을 제안한다: (i) `BLEND`: BLE의 도움을 받아 사용자의 이동성이 있을 때, 빠른 Wi-Fi 핸드오프 시스템, (ii) `PYLON`: Wi-Fi와 BLE, 관성센서, 실내도면을 활용하여 사용자의 이동경로 예측 시스템, (iii) `FINISH`: BLE와 관성센서, 기압센서를 활용하여 실내 환경에서의 네비게이션 시스템.

먼저, BLE의 도움을 받아 빠른 Wi-Fi 핸드오프를 제공하는 시스템을 제안한다. 시스템 설계에 앞서, 스마트폰에서 빠른 Wi-Fi 핸드오프가 일어날 수 없는지에 대한 원인를 실험 및 소스코드 분석을 통해 파악한다. 스마트폰에서는 AP와 연결되어 있는 시점부터 Wi-Fi 스캐닝 동작을 수행하지 않도록 설계되어 있기 때문에 핸드오프가 필요한 시점을 파악하여 강제로 스캐닝을 수행하도록 해야한다. 이 과정에서

BLE의 광고패킷을 활용하여 주변 AP에 대한 정보를 Wi-Fi 스캐닝 없이 제공할 수 있고, 이렇게 얻은 정보를 바탕으로 주변 AP와 현재 연결되어 있는 AP의 성능비교를 통해 핸드오프를 결정하게 된다. 성능평가를 위해 두 가지의 다른 최신 스마트폰에 애플리케이션을 통해서 실험을 진행하였고, 그 결과 이동하면서 비디오를 시청할 때 최대 111%의 성능향상을 가져왔다.

다음으로, Wi-Fi와 BLE를 활용하여 실내 환경에서 사용자의 이동 경로를 예측하는 시스템을 제안한다. 최근 스마트폰에는 기본적으로 관성센서가 탑재되어 있기 때문에, 센서의 값의 분석을 통해서 사용자의 움직임을 예측할 수 있다. 이를 활용하여, 사용자의 도움 없이 사용자의 이동 경로를 파악할 수 있고 이동 중에 수집된 Wi-Fi와 BLE의 신호세기를 통해서 가상 공간을 생성할 수 있다. 생성된 가상의 공간을 실제 실내 도면의 도움을 통해서 가상의 공간이 실제 환경에서 어떤 위치인지 파악하고 실제 문을 랜드마크로 삼아 사용자의 위치를 보정하게 된다. 위와 같이, Wi-Fi와 BLE, 관성센서, 실내 도면을 활용하여 사용자의 도움 없이 사용자의 이동 경로를 파악한다. 그 결과, 종류가 다른 다섯 대의 안드로이드 스마트폰으로 실험하였을 때, 1.42 m의 오차를 가지고 사용자의 위치를 정확하게 예측할 수 있다.

마지막으로, 앞서 제안한 측위 시스템을 바탕으로, BLE와 관성센서, 기압센서를 활용하여 실내 환경에서의 네비게이션 시스템을 제안한다. 두 번째 연구의 측위 시스템은 단일 층에서 동작했다면 네비게이션의 경우 건물 전체 층에서 동작할 수 있어야 한다. 따라서, 스마트폰에 내장되어 있는 기압센서를 활용하여 층을 구별하고 여러 사용자의 정보를 수집하여 건물 전체에 대해 라디오 맵을 건설한다. 건설된 라디오 맵을 기반으로 사용자의 초기 위치를 빠르게 파악하고 사용자가 제공하는 목적지를 바탕으로 최적의 경로를 제시한다. 그 결과, 100% 정확도를 가지고 사용자의 위치를 파악하고 최단 경로를 제시하는 것을 실험적으로 보인다.

요약하자면, Wi-Fi 및 BLE와 같은 기반 시설과 스마트폰에 탑재되어 있는 다양한 센서를 활용하여 사용자의 편의성 및 경험을 향상시키는 시스템을 제안한다. 이를 상용 스마트폰에 구현하여 실험을 통해 성능을 검증한다. 결과적으로, 사용자의 경험을 향상시킬 수 있는 우수한 성능을 보여주는 것을 확인한다.

주요어: 무선통신, Wi-Fi, BLE, 스마트폰, 관성센서, 핸드오프, 실내측위, 실내 네비게이션
학번: 2014-22586

감사의 글

스물셋의 나이에 연구실에 입학하여 6년간의 대학원 생활을 마무리하고 사회로 나가고자 합니다. 이십대의 청춘을 바쳐 학문적으로 성장했을 뿐만 아니라 정신적으로도 성숙해져가는 시간이었던 것 같습니다. 일련의 과정들 속에서 성장한 제 자신을 느끼며 그 간의 학위 과정이 앞으로 있을 제 인생의 소중한 자산임을 자부합니다. 여전히 부족하지만, 학위 과정을 보람되게 하고 또 무사히 마칠 수 있도록 도움을 주신 많은 분들께 감사의 인사를 드리고자 합니다.

먼저, 지난 5년간 부족한 저를 지도해주신 최성현 전 교수님, 현 전무님께 감사의 말씀을 올립니다. 지난 학위 과정 동안 전무님께 노력의 중요성에 대해서 배울 수 있었습니다. 좌절과 방황의 시기에 나태해진 저를 격려해주시고 성장할 수 있도록 올바른 방향으로 인도해 주셔서 정말 감사합니다. 날카로운 통찰력으로 연구지도를 해주심과 동시에 인간적으로는 따뜻하셨던 전무님으로부터 전문가의 자세를 배울 수 있어서 영광이었습니다. 앞으로도 사회에 나가 전무님의 가르침대로 항상 노력하고 배우는 자세로 살아갈 것을 다짐합니다.

두 번째 지도교수님이신 박세웅 교수님께도 깊은 감사의 말씀을 드립니다. 새로이 연구실에 함께하게 되었음에도 불구하고 저의 연구에 지대한 관심을 가져주시고 아낌없이 지도해주셔서 감사합니다. 또한, 교수님께서 평소에 해주셨던 좋은 말씀을 통해 인생을 올바르게 살아가는 방법을 배울 수 있어서 영광이었습니다. 미래에도 교수님께서 지도해 주신대로 노력하며 살아갈 것을 다짐합니다.

저의 학위 논문의 심사위원장을 맡아주신 심병효 교수님과 바쁘신 일정 가운데에도 심사에 참석해 지도해 주신 오정석, 이경한, 주창희 교수님께도 깊은 감사의

말씀을 올립니다. 항상 학생들을 편하게 대해주시고 응원해주신 심병효 교수님께 감사의 말씀 드립니다. 조금은 다른 분야이지만 깊은 통찰력으로 고견을 주신 이경한 교수님께 감사드립니다. 심사기간 동안 완성도 있는 학위 논문을 위해서 지도해주신 이경한 교수님께 감사의 말씀을 드립니다. 아낌없는 조언으로 저의 논문에 부족한 부분을 채워주신 주창희 교수님께 감사드립니다.

긴 시간 동안 함께 생활한 전 멀티미디어 및 무선통신망 연구실 선후배분들과 동기들에게도 감사의 말씀을 드립니다. 연구실에서 함께 생활했던 졸업하신 이원보, 이옥환, 홍종우, 가순원, 유승민, 곽규환, 손위평, 신연철, 구종회, 김성원, 변성호, 박승일, 박태준, 윤강진, 이규진, 양창목 선배님들께 감사의 말씀 저합니다. 제가 졸업한 후에도 치열하게 노력하고 연구에 매진하고 있을 김지훈, 이재홍, 이기택, 곽철영, 황선욱, 이지환, 이주헌, 이강현, 허재원, 이진명에게도 감사의 말씀을 전합니다. 먼저 사회로 나가 최선을 다하고 있을 장민석, 김병준, 임수훈, 이경진에게 감사의 말씀을 전합니다. 같이 졸업 준비를 하면서 서로에게 힘이 되었던 손영욱, 김준석, 윤호영에게 감사의 말을 전합니다. 특히 연구실에 처음 입학한 순간부터 마지막까지 함께했던 신연철, 구종회, 이규진, 허재원에게 특별한 감사의 인사를 드립니다. 제가 처음 연구실에 들어왔을 때부터 연구하는 방법과 방향을 제시해주었던 신연철 선배님께 감사드립니다. 부족한 저를 이끌어 주시고 격려해주시는 선배님의 모습에 더욱 연구에 매진할 수 있었고 지금의 제가 있을 수 있었습니다. 묵묵히 자신의 일을 다하고 제 일도 본인의 일인 것처럼 열정적으로 도와준 구종회 선배, 아무것도 몰랐던 저에게 자신의 노하우를 아낌없이 알려주고 함께 성장해 나갔던 이규진 선배, 비상함으로 저의 연구에 많은 도움을 주었던 허재원에게 말로 다 할 수 없는 감사함을 전합니다.

같이 연구실에서 생활한 유비쿼터스 네트워크 연구실 선후배님들, 이현중, 허정륜, 정승범, 강병준, 홍승근, 최시영, 박원빈, 이명섭, 김홍찬, 서지환, 박종연, 박은정, 김정준, 박정준, 양진모, 이굳솔, 백숭우, 김영석, 최시현, 윤지석, 유용재, 류도균, 윤예린에게 감사의 말씀을 드립니다. 연구실에 새로 함께하게 되었음에도 모두 따뜻하게 맞아주셔서 감사합니다. 특히 1년 동안 방장으로서 연구실을 이끌어준 김홍찬, 황선욱에게 감사합니다.

대학원 생활에서 작은 일탈을 함께한 김선도, 이규진, 이기택, 임수훈에게도 감사의 인사를 드립니다. 또한, 매주 PSZ 멤버들과 운동장에서 함께 축구를 즐겼던 시간은 그 어느 때 보다 즐거웠습니다. 물심양면으로 PSZ를 지원해주시는 이상화 교수님, 지금의 팀이 있기까지 무한한 노력을 기울인 윤민영, 백두산, 최선을 다해 팀을 이끌어가는 하석현, 김호열, 또 매주 함께 땀 흘린 모든 멤버들에게 감사하다는 말씀을 전합니다.

끝없는 믿음과 깊은 사랑으로 언제나 제 인생의 든든한 후원자가 되어 주시는 부모님께 이루 말할 수 없는 감사 인사를 드립니다. 항상 저를 믿어주시고 묵묵히 지원해주시는 두 분이 계시기에 지금의 제가 존재할 수 있었습니다. 저와 같은 길을 걸어가고 있는 자랑스러운 동생에게도 고맙습니다. 또한, 저를 친아들처럼 대해주시고 응원해주신 장모님께도 특별한 감사의 인사를 드립니다.

마지막으로, 누구보다 가까이서 저를 응원해주고 언제나 변함없이 제 옆자리를 지켜주는 아름다운 아내 지형이에게 사랑과 존경을 담아 감사의 인사를 드립니다. 힘들고 지칠 때 든든한 버팀목이 되어 주고, 따뜻한 마음과 지혜로운 생각으로 늘 귀감이 되는 아내와 함께 할 수 있다는 것이 앞으로 펼쳐질 제 인생에 정말 큰 행운이자 축복입니다. 항상 감사하고 사랑합니다.

모든 분들께 다시 한번 감사의 말씀을 올리며 이 논문을 바칩니다.

2020년 8월,

최준영 드림.